universität wien

DISSERTATION | DOCTORAL THESIS

Titel | Title

From Bit Flips to Lottery Tickets:
Expressivity in Graph Neural Networks

verfasst von | submitted by

Lorenz Kummer BSc MSc

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Doktor der technischen Wissenschaften (Dr.techn.)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 786 880
Dissertationsgebiet lt. Studienblatt | Field of study as it appears on the student record sheet:	Informatik
Betreut von | Supervisor:	Assoz. Prof. Dipl.-Inf. Dr. Nils Morten Kriege Univ.-Prof. Dipl.-Ing. Dr. Wilfried Gansterer M.Sc.

Acknowledgements

Deciding to pursue a PhD was one of the most exciting and rewarding choices of my life. These years have been full of discovery, challenge, and growth, both in knowledge and personally. I am deeply grateful to all who made this journey such a fun and enriching experience.

First and foremost, I would like to thank my supervisor, Nils Kriege, for his continuous support, guidance, and encouragement throughout my PhD. You gave me the freedom to pursue my own ideas while always being available for discussions and invaluable feedback. I also owe special thanks to my co-supervisor, Wilfried Gansterer. Working as a teaching and research assistant with you during my master's studies first sparked my desire to pursue a PhD, and I am grateful for your trust and support ever since.

I also want to thank my co-authors for the fruitful collaborations that shaped this thesis: Samir Moustafa, Sebastian Schrittwieser, Franka Bause, and Anatol Ehrlich. Special thanks go to Tabea Reichmann, Kevin Sidak, and Nikolaus Süß. With you, collaboration went beyond co-authorship, as you also became close friends who made these years much more enjoyable.

I am equally grateful to my fellow PhD students in the office: Gaith, Timo, Martin, Simon, Gabriele, Joao, and again Franka and Anatol. Thank you for the countless discussions, shared coffee breaks, and the good atmosphere we had together. You made everyday PhD life a truly good time.

During my PhD I had the privilege of presenting my work at conferences around the world. These opportunities allowed me to share ideas, meet inspiring researchers, and gain feedback that shaped my work. I fondly remember my discussions with Przemysław Wałęga at AAAI in Philadelphia and with Shahaf Finder at ICML in Vancouver, exchanges that broadened my perspective and encouraged me to refine my ideas further.

Beyond academia, I am thankful for my friends who supported me through these years: Thomas, Luijza, Marco, David, Michael, Valentyn, and Melanie. In particular, I want to thank Valentyn, who helped me stay motivated during challenging times, and Melanie, who, almost a decade ago, encouraged me to go to university in the first place.

Most importantly, I thank my family. To my parents, Christian and Michaela, thank you for your constant support and for always being there for me. And to my wife Gabi, and my daughters Emma and Nora, you are the center of my life. I cannot thank you enough for your love, patience, and for enriching every day anew. My greatest wish is to see our daughters grow up together with you by my side.

Looking back, I truly enjoyed my PhD studies. They were challenging, fun, and exciting, and I consider myself very lucky to have had the chance to experience this journey.

Bibliographical Note

The chapters of this thesis are based on the following four papers, which have been published in conference proceedings:

Paper 1 Lorenz Kummer, Samir Moustafa, Sebastian Schrittwieser, Wilfried Gansterer, Nils Kriege. "Attacking Graph Neural Networks with Bit Flips: Weisfeiler and Leman Go Indifferent". In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2024

Paper 2 Lorenz Kummer, Samir Moustafa, Wilfried Gansterer, Nils Kriege. "Crossfire: An Elastic Defense Framework for Graph Neural Networks Under Bit Flip Attacks". In: Proceedings of the AAAI Conference on Artificial Intelligence, 2025

Paper 3 Lorenz Kummer, Wilfried Gansterer, Nils Kriege. "On the Relationship Between Robustness and Expressivity of Graph Neural Networks". In: Proceedings of the 28th International Conference on Artificial Intelligence and Statistic, 2025

Paper 4 Lorenz Kummer, Samir Moustafa, Anatol Ehrlich, Franka Bause, Nikolaus Suess, Wilfried Gansterer, Nils Kriege. "Weisfeiler and Leman Go Gambling: Why Expressive Lottery Tickets Win". In: Proceedings of the 42nd International Conference on Machine Learning, 2025

Papers (co)authored during the PhD but not included in this thesis:

Paper 5 Lorenz Kummer, Kevin Sidak, Tabea Reichmann, and Wilfried Gansterer. "Adaptive Precision Training (AdaPT): A dynamic quantized training approach for DNNs". In: Proceedings of the SIAM International Conference on Data Mining, 2023

Paper 6 Samir Moustafa, Lorenz Kummer, Nils Kriege, Wilfried Gansterer. "Visualization and Analysis of the Loss Landscape in Graph Neural Networks". In: Proceedings of the 34th International Conference on Artificial Neural Networks, 2025

Paper 7 Anatol Ehrlich, Lorenz Kummer, Vojtech Voracek, Franka Bause, Nils Kriege. "XIMP: Cross Graph Inter-Message Passing for Molecular Property Prediction". Submitted for review at the International Conference on Learning Representations, 2026

Abstract

Complex objects such as molecules, proteins, or social networks can be naturally represented as graphs, and Graph Neural Networks (GNNs) have emerged as powerful tools for learning on such graph-structured data. GNNs thereby extend the applicability of established deep learning techniques to new domains such as financial and medical data analysis, or chem- and bioinformatics, leading to scenarios where *efficient* deployment is crucial. This increasing adoption leads to new security challenges, particularly around the interplay between efficiency-enhancing techniques such as quantization and *expressivity*, a key property of GNNs denoting their ability to distinguish graph structures, as well as the *robustness* of GNNs to malicious activity. In this thesis, we focus on the robustness of GNNs to malicious perturbations of their parameters, specifically examining their vulnerability under assumptions on their numerical representation and precision. Moreover, we discuss the importance of expressivity in the context of sparse GNN parameterizations and show that careless sparsification can lead to irrecoverable loss of predictive performance.

We introduce the first Bit-Flip Attack (BFA) dedicated to maliciously perturbing the binary representation of the parameters of a quantized GNN. BFAs have so far been almost exclusively studied for Convolutional Neural Networks (CNNs). Our BFA is specifically designed to degrade a GNN's ability to impair expressivity, highlighting that this fundamental property can be leveraged as a pivotal point by an attacker.

Consequentially recognizing the importance of defending GNNs from BFAs, we offer the first study on how well existing defenses transferred from CNNs can protect quantized GNNs. We build upon the insights from these results and propose Crossfire. Crossfire is not only the first BFA defense dedicated to GNNs, but also the first BFA defense in general seeking to not only maintain the predictive performance of a target model or detect an attack, but to also verifiably restore the network to its pre-attack state.

To further deepen our understanding of the robustness-expressivity relationship, we examine the vulnerability of a certain class of GNNs and establish formal criteria to characterize a GNN's susceptibility to losing expressivity due to BFAs in a first theoretical analysis. This enables an analysis of the impact of homophily, graph structural variety, feature encoding, and activation functions on GNN robustness, leading to the discovery of particularly vulnerable configurations. Moreover, we derive theoretical bounds for the number of bit flips required to degrade GNN expressivity on a given dataset.

Going beyond quantization to enhance efficiency and extending the established Lottery Ticket Hypothesis with new insights for GNNs, we identify expressivity as crucial for finding sparse initializations (*winning tickets*) that preserve the predictive performance. We establish conditions under which the expressivity of a sparsely initialized GNN matches that of the full network. Moreover, we subsequently show that an increased expressivity in the initialization potentially accelerates model convergence and improves generalization.

Kurzfassung

Komplexe Objekte wie Moleküle, Proteine oder soziale Netzwerke lassen sich natürlich als Graphen modellieren, für welche sich Graph Neural Networks (GNNs) als leistungsstarkes Lernverfahren etabliert haben. GNNs übertragen bewährte Deep-Learning-Techniken auf Bereiche wie die Analyse von Finanz- und Sozialnetzwerken oder medizinischen Daten sowie Chemie- und Bioinformatik und führen so zu Anwendungen, in denen *Effizienz* zentral ist. Diese zunehmende Verbreitung macht es notwendig, potenzielle Sicherheitsrisiken zu erforschen, insbesondere das Zusammenspiel von *Expressivität*, welche die Fähigkeit von GNNs bezeichnet, Graphstrukturen zu unterscheiden, und effizienzsteigernden Verfahren wie Quantisierung. In der vorliegenden Dissertation untersuchen wir die *Robustheit* von GNNs gegenüber gezielten Parameterstörungen und analysieren ihre Verwundbarkeit unter Annahmen über ihre numerische Darstellung und Präzision. Zudem diskutieren wir Expressivität im Kontext sparsifizierter GNN-Parametrisierungen und zeigen, dass unbedachte Sparsifizierung zu irreversiblen Einbußen der Vorhersageleistung führen kann.

Wir stellen den ersten Angriff vor, der die binäre Darstellung quantisierter GNNs mittels Bit-Flip-Angriffen (BFAs) gezielt manipuliert, wobei es sich um ein Verfahren handelt, das bisher überwiegend für Convolutional Neural Networks (CNNs) untersucht wurde. Unser BFA zielt explizit darauf ab, die Expressivität eines GNN und damit seine Fähigkeit zur Unterscheidung von Graphstrukturen zu beeinträchtigen und zeigt damit, dass ein Angreifer diese fundamentale Eigenschaft ausnutzen kann.

Angesichts dessen untersuchen wir erstmals, wie gut bestehende, von CNNs übertragene Abwehrmechanismen GNNs schützen. Basierend auf diesen Erkenntnissen entwickeln wir Crossfire, die erste speziell für quantisierte GNNs entwickelte BFA-Abwehr und auch die Erste, die nicht nur die Vorhersageleistung des Zielmodells erhält oder einen Angriff erkennt, sondern das Modell nachweislich in den Ursprungszustand zurückversetzen kann.

Um das Bild zu vervollständigen, analysieren wir die Verwundbarkeit einer bestimmten Klasse von GNNs und entwickeln erste formale Kriterien, um die Anfälligkeit für Expressivitätsverluste durch BFAs zu charakterisieren. Dies ermöglicht eine Analyse des Einflusses von Homophilie, struktureller Vielfalt, Merkmalskodierung und Aktivierungsfunktionen auf die Robustheit von GNNs und führt zur Identifikation besonders verwundbarer Konfigurationen. Überdies leiten wir Schranken für die Anzahl an Bit-Flips ab, die nötig sind, um die Expressivität eines GNN auf einem Datensatz zu verringern.

Über Quantisierung hinaus identifizieren wir die Expressivität sparsifizierter Subnetzwerke als Schlüssel, um sparsifizierte Initialisierungen („Winning Tickets“) zu finden, die die Vorhersageleistung erhalten. Wir geben Bedingungen an, unter denen die Expressivität eines derart initialisierten GNN jener des vollständigen Netzwerks entspricht. Zudem zeigen wir, dass höhere Expressivität in der Initialisierung potenziell Konvergenzverhalten und Generalisierungfähigkeit verbessert.

Contents

1. Introduction

Graphs provide a flexible mathematical representation for encoding complex entities, ranging from molecules and proteins to social and information networks, where nodes stand for the constituent parts, edges capture their relationships, and both nodes and edges can be enriched with domain-specific feature vectors.

Building on this representation, Graph Neural Networks (GNNs) offer a powerful learning paradigm for graph-structured data. Among others, a key concept in GNN research is expressivity, which refers to a GNN's ability to distinguish graphs for which no edge-preserving bijection between their node sets exists. Such graphs are called non-isomorphic. A minimal requirement for the learnability of a task by a GNN is that the GNN can separate at least those non-isomorphic graphs that the task distinguishes through, for example, different labels or contrastive relations. More generally, the ability of GNNs to distinguish non-isomorphic graphs has been linked to their ability to approximate universal functions on graphs [CVCB19, BG20]. Expressivity is often benchmarked via the Weisfeiler-Leman (WL) isomorphism test, with the 1-dimensional WL (1-WL) test being a standard baseline for many GNNs [WL68, MLM$^+$23]. On a more fine grained level, [Lou20] show that whether certain GNNs can decide nontrivial graph properties depends on their depth (number of layers) and width (number of parameters per layer) and [AGA$^+$23] discuss the impact of activations on expressivity. Yet, these results largely operate on an abstract functional level, which is the prevailing perspective in the literature on expressivity: parameters are typically treated as real-valued and the role of individual parameters or even bits is not examined.

Beyond the branch of research that has focused on theoretical aspects such as expressivity, GNNs have also proven highly effective in practical settings. They have extended the applicability of neural networks (NNs) to various new fields, such as financial and social network analysis, medical data analysis, and chem- and bioinformatics [GLX$^+$22, LU21, SYC$^+$21, XXC$^+$21, CM20, WRF$^+$18]. Hence, as GNNs see growing adoption across these critical domains, it becomes increasingly important to examine their potential security vulnerabilities.

Such security vulnerabilities can come in various forms. Conventional adversarial attacks on GNNs primarily target the input data [WCP$^+$22], either through poisoning attacks, where manipulated training data leads to a compromised model [WCP$^+$22, MDM20], or through evasion attacks, which introduce adversarial examples to impair inference performance. These attacks can be targeted [ZAG18] or untargeted [MDM20, ZG19] and, in the context of GNNs, involve altering node features, modifying edge structures, or injecting new nodes into the input graphs [WCP$^+$22, SWT$^+$20]. Targeted attacks focus on disrupting predictions for a selected subset of instances, whereas untargeted attacks aim to broadly impair the model's predictive performance [ZCS$^+$22]. For an overview of

graph poisoning and evasion attacks, as well as associated defenses and a repository with representative algorithms, see the comprehensive reviews by [JLX+21] and [DZZ+22].

Beyond these data-centric threats, a separate class of attacks targets the model parameters directly. Previous research on the security vulnerabilities of convolutional neural networks (CNNs), which are prominent in the computer vision domain, has revealed that CNNs are highly susceptible to malicious Bit-Flip Attacks (BFAs) [HFK+19, QZN+23]. Orthogonal to poisoning and evasion attacks, which operate by perturbing the input data, BFAs compromise the model directly by altering the binary in-memory representation of individual trainable parameters via perturbations of selected bits. BFAs typically employ gradient-based bit-search schemes to determine the most vulnerable bits in a target model. The majority of these works focus on CNNs to which integer quantization has been applied, for example, [RHF19, QZN+23], a common technique to improve efficiency. Integer quantized CNNs are naturally more perturbation resilient compared to their unquantized (that is, floating-point) counterparts [RHL+22, HFK+19], and thus their degradation poses a more challenging problem. For example, quantization is even mentioned by [HFK+19] as a viable protection mechanism against random BFAs.

Going beyond merely changing the numerical representation to an inherently more robust format, a variety of defensive mechanisms have been proposed for CNNs. Major trends in this area are network hardening, detection, and proactive restoration. Network hardening techniques focus on preventing successful BFAs either via learned robustness [HRL+20, LRX+20, CMDLP23] or deflecting an attacker's bit-search scheme [ZWC+22, WZW+23], but may require (re)training and do not repair compromised neurons. Detection-based defenses [JK21, LRH+21, OL22] identify bit flips efficiently and can be applied post hoc, yet offer limited restoration. Proactive methods [LYW+23, LRX+20] anticipate attack patterns and emphasize weight restoration, either through reverting or statistical correction. For a comprehensive overview, see [KLI+22].

Given the observed interaction between robustness to BFAs and efficiency in quantized CNNs, it is natural to ask how these effects manifest in the context of GNNs. The practical relevance of quantizing GNNs has been recognized for applications such as inference on the Internet of Things (IoT) and edge devices [YZY+22], reducing energy consumption for green deep learning [XZF+21], and in distributed GNN applications [SLG+24]. Recent efforts have been made to further advance quantization techniques for GNNs [MKG25, ZLM+23, BBZ21, FWL+20] as well as technical realizations for their deployment [ZLM+23, SJN21] and further potential applications exist, for example, [DTW+23, BF23, DPSW+21]. However, in contrast to poisoning and evasion, research on the resilience of quantized and floating point GNNs to BFAs has received surprisingly little attention. At the time of writing, and to the best of our knowledge, only two prior studies aside from our own have considered this vulnerability [JWL+22, WYW+24]. Specifically, [JWL+22] only empirically study random bit faults in GNNs, and [WYW+23] adapt a classical attack [RHF19] for CNNs to flip exponent bits in GNN parameters' floating-point representations. Yet, while expressivity is fundamental to understanding and designing GNNs, its potential role in the robustness of such models has so far been entirely disregarded in the literature.

Quantization is, however, not the only means to increase the efficiency of NNs. Another such method is pruning or sparsification, which can be applied before, during, or after training. An influential concept regarding pre-training sparsification (that is, pruning before training, at initialization time) is the Lottery Ticket Hypothesis (LTH), originally proposed by [FC18]. LTH posits that large, randomly initialized NNs contain smaller, trainable subnetworks, or *winning tickets*, that can match the prediction quality of the dense network. LTH has attracted considerable attention, particularly in the context of deep learning, where these subnetworks are typically identified via iterative pruning techniques, significantly reducing model size and computational cost while preserving predictive performance. Going beyond LTH, the formally proven Strong Lottery Ticket Hypothesis (SLTH) establishes that over-parameterized NNs contain subnetworks that achieve a high degree of prediction quality even without training [MYSSS20]. LTH has since been extended to various domains, including GNNs. In the context of GNNs, LTH has been investigated mainly through pruning both graph structures (such as adjacency matrices) and model parameters (that is, the weights of transformation matrices), resulting in the discovery of Graph Lottery Tickets (GLTs) that preserve prediction quality while substantially lowering computational costs [CSC+21, TP23]. A comprehensive survey on LTH and related works is provided [LZH+24]. Remarkably, the relationship between LTH and GNN expressivity has not been analyzed formally so far, and whether pruning as an efficiency-enhancing approach orthogonal to quantization interacts with GNN robustness to BFAs has likewise not yet been explored.

In this thesis, we hence focus on addressing this research gap by exploring the effects of malicious bit-level perturbations of the trainable parameters of (quantized) GNNs as well as their impact on prediction quality as well as countermeasures. Specifically, we do so under the consideration of GNN expressivity, that is, their ability to distinguish non-isomorphic graphs or node neighborhoods. Moreover, extending our research beyond quantization to gain efficiency, we explore the interaction between expressivity and LTH, which provides a stepping stone for future exploration of the robustness of sparsified GNNs to BFAs. In doing so, we step away from the abstract functional perspective on expressivity common in literature and provide a fine grained view of the roles that individual parameters and bits can play in a GNN's capability to distinguish non-isomorphic graphs. Our unique perspective allows us to gain novel insights into the interplay between efficiency, expressivity, and robustness.

1.1. Research Goals

The overarching goal of the studies composing this thesis was to address selected problems at the intersection of three key aspects of GNNs – efficiency, expressivity, and robustness – as well as their interactions. Figure 1.1 illustrates how the subproblems integrate into the triad, whereby each subproblem is represented by its corresponding paper. As addressing all aspects of these interactions conclusively was clearly beyond the thesis's scope, we focused on a numerical perspective for much of the thesis and on specific, highly relevant subproblems.

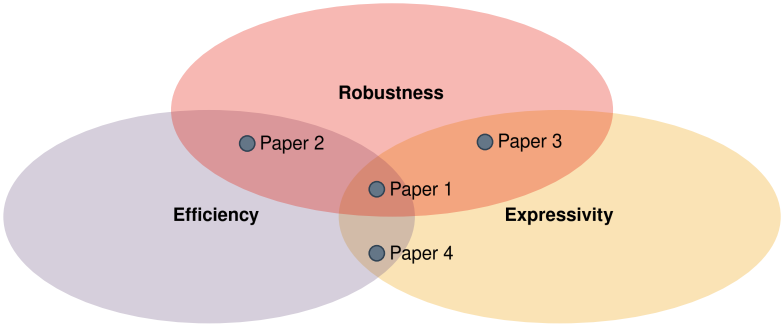


Figure 1.1.: Venn diagram showing how the four contributed papers (1-4) address sub-problems at the intersection of efficiency, robustness, and expressivity in GNNs, and how each maps onto these three aspects.

The first such sub-problem (Paper 1) was to determine whether the known robustness of integer quantized CNNs to BFAs [RHL+22, HFK+19] transfers to GNNs as well, as it is mostly rooted in the numerical representation itself, and how the unique mathematical properties of GNNs that give them their expressivity might be exploited by an attacker. Particularly in light of results that showed that adapting BFAs to the specific properties of a target architecture can increase harm [VMA+20] and that BFAs can be far from optimal on non-convolutional models [HMDD22], we found it crucial to improve our understanding of the relationship between GNN expressivity robustness against BFAs.

The second sub-problem (Paper 2) we studied was motivated by the observation that existing defenses against BFAs face shared challenges and limitations. First, they do not include mechanisms to guarantee complete network restoration following an attack; although prediction quality may be restored, the overall integrity of the network typically remains uncertain. This is undesirable, as reloading a clean model or switching to a backup system after detecting an attack is often inefficient and introduces delays [LRH+21]. Second, depending on a single defense mechanism makes these approaches vulnerable to BFAs that exploit specific loopholes [LYW+23]. Moreover, it remains unclear whether defense mechanisms designed for CNNs can be effectively transferred to GNNs. Considering the critical roles of GNNs in applications such as medical diagnosis [LQZL20, LU21], health record modeling [LLP+20, SYC+21], and drug development [XXC+21, CM20], ensuring their authenticity post-attack is essential. Our goal, hence, was to develop a robust defense strategy that integrates multiple protective layers and incorporates efficient, low-cost verification.

The third sub-problem (Paper 3) was motivated by recent results indicating that GNN expressivity is generally resilient to random bit flips [AGA+23]. Given the key role of expressivity in GNNs, we found it necessary to extend upon our study of the first sub-problem and provide a theoretical analysis of the resilience of GNNs to malicious bit flips. Specifically, we aimed at delivering theoretical bounds on the number of malicious bit flips a GNN may withstand and at analyzing how these bounds change with varying assumptions on the input graphs' structure and the GNN architecture, thus allowing us

to identify more and less robust configurations.

The fourth and final sub-problem (Paper 4) was based on the observation that, despite extensive work on LTH in GNNs, the link between LTH and GNN expressivity is unexplored. While prior research focused on efficiency, the effect of pre-training pruning on preserving expressivity was largely overlooked. Our goal, hence, was to understand this connection and reveal how sparse initialization impacts a GNN's ability to distinguish complex graph structures. The long-term aim of addressing this gap is to set the foundation for the development of more efficient graph learning models that maintain expressivity.

The research goals and associated sub-problems outlined above serve as the foundation for the chapters that follow. An overview of these chapters is provided below.

1.2. Thesis Structure

This thesis is organized as follows. Chapter 2 provides the necessary background relevant to the contributions presented in this work. Chapter 3 outlines our main contributions in detail, highlights the commonalities among the conducted studies and proposed methods, and explicitly outlines how they distinguish themselves from related works. Chapter 4 concludes the thesis with a discussion of the studies and methods, their limitations, and potential directions for future research. Appendix A includes the four published papers that constitute this thesis, as well as a detailed breakdown of co-author contributions.

2. Background

The contributions of this thesis are based on existing research on GNNs, BFAs, and LTH. In this chapter, we hence provide the general background information necessary for the subsequent discourse on the contributions of this thesis. As the papers associated with this thesis are already self-contained and included in Appendix A, we only discuss each field briefly. Where necessary for understanding, we will include brief, technical footnotes on specific key details. The interested reader will find detailed references in each section, guiding them to more related research.

2.1. Graphs

Graphs are a mathematical construct used to model the relationship between entities, whereby each entity is represented by a node and each relationship between two entities is represented by an edge. Both nodes and edges can be endowed with feature vectors. While graphs with directed edges (so-called directed graphs) [WR25] or graphs with edges connecting more than two nodes (so-called hypergraphs) [HY21, WSZ+22] exist and are a subject of ongoing research, we only consider undirected graphs and edges modeling binary relationships in the following, as they are highly common and relevant in science due to their dominant place in common graph learning benchmark datasets [MKB+20, HFZ+20, xO25a, xO25b, WRF+18]. Moreover, we limit our discussion to homogeneous graphs, that is, graphs of only a single node and edge type, such as atoms and their bonds as found in molecular property prediction, respectively. Heterogeneous graphs can be composed of multiple node and edge types, but as these typically have different semantic meanings (such as user and item, as found in recommendation systems [WSZ+22]) and can have different length feature vectors, these different node and edge types usually require specialized architectural adaptations and notions of expressivity [BGMO22]. Another constraint of the following discussion is its focus on static graphs. Opposed to dynamic graphs, which can change over time [WR25] (as, for example, found in social networks [MGP+21]), the assumption of static graphs is that their structure and composition remain stable over time. Note, though, that certain ideas developed in the context of this thesis are likely extendable beyond these constraints, as we shall argue in Section 4.3 (Future Work).

2.2. Graph Neural Networks

GNNs leverage graph structure and node features to derive representation vectors for specific nodes or graphs. Contemporary GNNs typically use a neighborhood aggregation or message-passing (MP) approach [MLM+23], where the representation of a node is

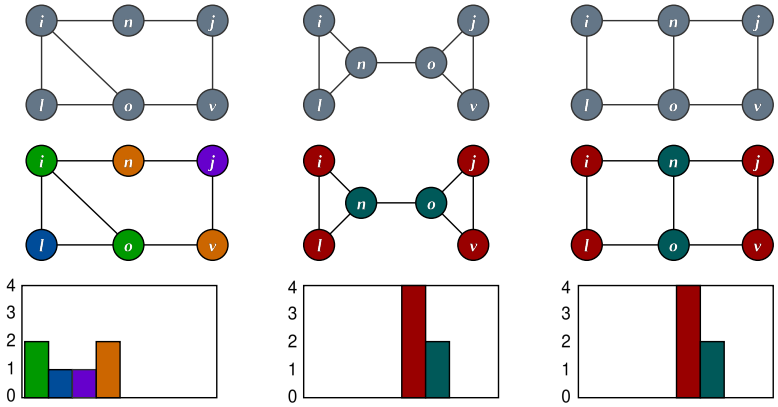


Figure 2.1.: Three pairwise non-isomorphic graphs (left, center, right). Pairs (left, center) and (left, right) are distinguished by the 1-WL test; pair (center, right) is not. Top row: initial homogeneous node labels. Middle row: node colors after six rounds of color refinement. Bottom row: corresponding color histograms.

iteratively updated by first aggregating the representations of its neighboring nodes (see Figure 2.2) and then applying a learnable non-linear transformation (typically a feedforward network, such as a Multilayered Perceptron (MLP)). This aggregation step can be viewed as learning a permutation-invariant function over a multiset of neighboring node features and, for example, be realized by the Deep Sets framework [ZKR+17] and its extensions to multisets that account for both the identity and multiplicity of elements [XHLJ19, AGA+23]. For obtaining graph level representations, individual node embeddings are aggregated into a single representation vector. Hence, the node and representations generated by GNNs are representations of their local structure and features [XHLJ19, WK16].

2.2.1. Expressivity

The concept of expressivity can, in the widest sense, be defined as the set of functions a machine learning model can approximate. In graph learning, though, it is commonly defined as the ability of an algorithm or machine learning model to distinguish structurally different graphs or substructures of graphs. While the connection between the two definitions has been subject to research [CVCB19, BG20], we will subsequently mean the latter one whenever we mention expressivity.

The Weisfeiler-Leman Test Typically, expressivity is benchmarked against the 1-WL test, a color refinement heuristic for testing whether two graphs are isomorphic, that is, structurally indistinguishable[1]. Despite its long history, the WL test remains an active area

[1]Formally, two graphs are isomorphic if there exists a bijection between their nodes that preserves which nodes are adjacent; non-isomorphic graphs admit no such bijection [WL68]. If nodes or edges have labels, the bijection must preserve those as well [BGMO22].

of research [Kie20]. While many more expressive variants of the WL algorithm exist [Gro21, BGMO22], as well as GNN architectures built upon them [MRF+19, BGMO22], 1-WL still distinguishes most graphs in typical benchmark datasets [MFK21, Zop22]; hence we discuss only this variant and corresponding GNN architectures.

The 1-WL algorithm operates by iteratively refining a coloring of the two graphs' nodes based on their local structural properties. The algorithm typically starts by assigning initial colorings to each node, based, for example, on its degree or other properties. In each iteration, the algorithm then updates each node's color by injectively hashing the colors of a given node together with the multiset of the colors of its neighbors, which are the nodes adjacent to the given node. This step is repeated until either both graph's colorings stabilize or their color histograms differ, whereby in the latter case, the algorithm decides that the two graphs are non-isomorphic [Gro21]. A conceptual visualization is provided in Figure 2.1.

Expressivity in GNNs While GNNs corresponding to aforementioned extensions of the WL framework beyond 1-WL exist, such as relational [BGMO22] and higher-order variants [MRF+19, MLM+23], extending our analysis to these cases is beyond the scope of this thesis and left for future work (see Section 4.3). Accordingly, we restrict the following discussion to architectures whose expressivity is bounded by the 1-WL test.

Within this setting, Graph Isomorphism Network (GIN) [XHLJ19], which was explicitly designed to match the expressivity of the 1-WL test, set a new standard in graph representation learning as its simple but powerful aggregation function allows it to distinguish all non-isomorphic graphs that are also distinguishable by the 1-WL test[2]. Together, these contributions have shaped our current understanding of GNN expressivity, where models are assessed by their capacity to approximate or exceed the distinguishing capabilities of the 1-WL test on complex graph structures. A substantial body of research has been devoted to extending GNN expressivity beyond the inherent limitations of 1-WL and increasing the model's capacity to distinguish between structurally distinct graphs [MLM+23]. Despite these advances, the 1-WL test continues to distinguish most graphs in commonly used benchmarks [MFK21, Zop22], indicating that exceeding this foundational approach is not necessary in many scenarios.

The practical impact of model expressivity on prediction quality is not entirely clear [FMVG24]. However, for example, in molecular property prediction and drug discovery, more expressive GNN architectures can, in certain instances, be associated with improvements in prediction quality [FYW20, WKH+24]. Generally, structurally similar molecules can exhibit substantially different potencies against the same protein target, a phenomenon referred to as activity cliffs [PVMLSAMF15]. Accurately identifying activity cliffs – that is, making fine-grained structural distinctions among molecular graphs –

[2]GIN's expressivity stems from its use of injective multiset aggregation (for example, summation) combined with an injective MLP. This design ensures that distinct neighborhood multisets are mapped to distinct embeddings, thereby enabling GIN to match the distinguishing power of the 1-WL test [XHLJ19]. If the aggregation or the transformation is non-injective, different structures may be mapped to the same representation, reducing expressivity.

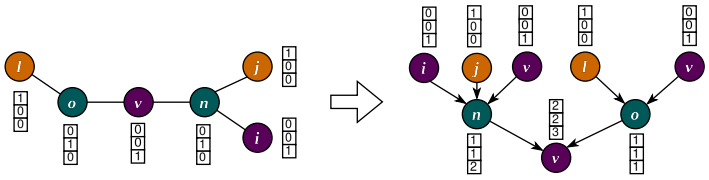


Figure 2.2.: Conceptual visualization of message passing in a GNN. Left: a graph with node feature vectors given by one-hot encodings of the node labels (colors). Right: sum aggregation of messages from the 2-hop neighborhood of node v, where the final representation of v reflects both its structural neighborhood and the associated feature information. For clarity, non-linear transformations typically applied after each round of aggregation are omitted.

is both challenging and critical, as misclassification can lead to drug candidates being mistakenly assessed as either safe or toxic. Furthermore, distinguishing stereoisomers is crucial, as they exhibit identical molecular formulas and bond structures, differing only in their three-dimensional atomic arrangement. Even such subtle differences can significantly alter a molecule's biological activity and pharmacological profile, an issue that will resurface in our discussion of the connection between LTH and expressivity.

2.2.2. Architectures

While some MP-GNN architectures, such as GIN, are specifically designed to improve the ability to distinguish non-isomorphic graphs, others, such as the Graph Convolutional Network [WK16], were originally derived from a spectral graph convolution perspective and encourage smoothing of node features over the graph. In the message-passing framework, GCN falls into the class of degree-aware GNNs, whose distinguishing power is closely tied to 1-WL, whereby the original variant is strictly less expressive than 1-WL and only attains full 1-WL power after a slight extension with a learnable trade-off parameter between features of the vertex and those of its neighbours [GMP21]. Graph Attention Networks [VCC+18] (GAT), in turn, introduce attention mechanisms to dynamically weight neighboring nodes, thereby improving the flexibility of feature aggregation without explicitly targeting higher expressivity. In practice, GIN often excels on graph-level tasks that require fine structural discrimination (such as, for example, molecular property prediction [HFZ+20]), whereas GCN performs strongly on homophilious node-classification benchmarks due to its smoothing behavior [RJK+20]. All three (GIN, GCN and GAT), though, remain bounded in their expressivity by the 1-WL test, unless special preprocessing steps, such as node feature augmentations [AGKG22], are used.

2.3. Bit-Flip Attacks

Bit-Flip Attacks, or BFAs, in general are inference time attacks on the binary representations of a model's trainable parameters. Similar to the distinction made between

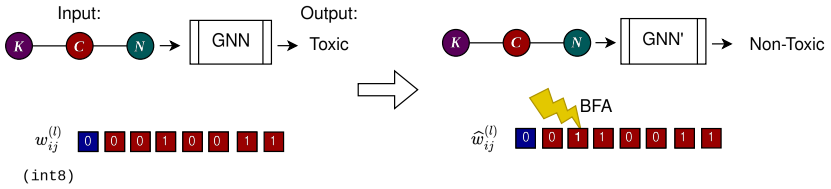


Figure 2.3.: Conceptual visualization of a BFA on the binary representation of an `int8`-quantized GNN's weight element $w_{ij}^{(l)}$, whereby l indicates the layer and ij the position in the weight matrix. In the example, the molecular graph representing potassium cyanide (with node labels constituting atom types) is correctly classified as toxic before the attack but misclassified as non-toxic after the attack. We discuss quantization in detail in Section 2.3.2.

targeted [ZAG18] and untargeted [MDM20, ZG19] poisoning and evasion techniques, [QZN+23] and [KLI+22] distinguish between targeted and untargeted BFAs.

2.3.1. Fundamental Assumptions and Threat Model

BFAs employ bit-search algorithms (detailed in Section 2.3.3) that seek to find exactly those bits that, if flipped (that is $1 \rightarrow 0$ or $0 \rightarrow 1$) will adapt the model best to suit the attackers' goal, and then realize those bit flips in hardware through mechanisms such as RowHammer [MK19][3], variants thereof [YRF20, LSR+20] or others [HBJ+20, BHJ+18]. Hence, the underlying assumption of BFAs is that the attacker has the capability to exactly flip the bits chosen by the bit-search algorithm; an assumption for which feasibility was shown by, for example, [WZW+23], who induced exact bit flips via a RowHammer variant. This assumption is reflected in the gray-box threat model[4] that is typically assumed for BFAs in literature. This setting relaxes some of the limitations of a pure black-box model by granting the attacker partial knowledge of the training data and the model architecture [LYW+23]. Such information is often easily accessible to an attacker, for example, when the target model is a publicly available pre-trained network or when the attacker can exploit an identical instance of a device on which model inference is performed. Even in the absence of explicit prior knowledge, attackers may still extract model parameters and input data via side-channel attacks [YFT20, BBJP18]. Figure 2.3 provides a conceptual illustration of a BFA on a GNN. Another typical assumption in the related work on BFAs is that the target network is integer quantized. A large body of work focuses on BFAs for quantized CNNs [QZN+23, KLI+22] and associated BFA

[3] RowHammer is a hardware fault injection technique that exploits charge leakage in DRAM. By repeatedly accessing ("hammering") memory rows adjacent to a target row at high frequency, it induces electrical disturbance that can flip bits in the intervening (victim) row, despite it not being accessed directly.

[4] An intermediate threat model: the attacker has partial knowledge of the target model, but not full access to the parameters, training data or hyperparameters. This contrasts with the white-box threat models, which assume complete access, and the black-box threat models, which allow only input–output queries with no access to internal states or training data.

defense mechanisms, for example, [KLI+22, LRH+21, LYW+23], as integer quantized NNs are generally more perturbation resilient than their corresponding floating-point counterparts, as we shall detail soon. This typically makes them a more challenging target for an attacker. Similarly, integer-quantized NNs are of interest from the defender's perspective because common quantization schemes (for example, scale quantization; see below) make bit-level perturbations of individual weights harder to detect, since simple out-of-distribution (OOD) checks alone may not suffice.

2.3.2. Quantization

Quantization involves decreasing the numerical precision of weights, biases, and activations, or adopting more compact representations, which in turn reduces the overall model size and memory footprint.

While other and more GNN specific quantization methods exist [SJN21, MKG25], the typical set-up chosen in the related work on BFA on CNNs [RHF19, HRL+20][5] is to apply scale quantization to map `float32`[6] tensors to the `int8` range (with a visualization of both binary formats provided in Figure 2.4). An appropriate quantization function $\mathcal{Q}$ and its associated dequantization function $\mathcal{Q}^{-1}$ are given by

$$\begin{aligned} \mathcal{Q}(\mathbf{W}^{(l)}) &= \mathbf{W}_q^{(l)} = \text{clip}(\lfloor \mathbf{W}^{(l)}/s \rceil,\ a, b), \\ \mathcal{Q}^{-1}(\mathbf{W}_q^{(l)}) &= \widehat{\mathbf{W}}^{(l)} = \mathbf{W}_q^{(l)} \times s. \end{aligned} \tag{2.1}$$

Here, the variable s denotes the scaling parameter, $\text{clip}(x, a, b) = \min(\max(x, a), b)$ defines the clipping operation with a and b representing the minimum and maximum thresholds (also referred to as the quantization range), $\lfloor \dots \rceil$ denotes rounding to the nearest integer, $\mathbf{W}^{(l)}$ is the weight matrix of the layer l to be quantized, $\mathbf{W}_q^{(l)}$ is its quantized counterpart, and $\widehat{\dots}$ indicates a perturbation. Specifically, rounding errors in the case of Equation (2.1).

Typically, works on BFA that require quantized target networks, for example, [RHF19], address the issue of non-differentiable rounding and clipping functions in $\mathcal{Q}$ by employing Straight-Through Estimation (STE)[7] [BLC13].

The inherent robustness of quantized CNNs to perturbations is largely due to their numerical representation. Integer-based quantization restricts the impact of single bit flips, limiting how much individual parameters can change. For instance, in the 8-bit integer two's complement representation, the maximum possible change from a single bit flip is 128, whereas a flip in the exponent of an IEEE 754 32-bit floating-point number

[5] This detail is not stated explicitly in the papers; it is evident from the implementation in the authors' GitHub repository (`https://github.com/elliothe/BFA`).

[6] Typically, floating-point numbers (for example, `float32`) follow IEEE 754 (`https://754r.ucbtest.org/`) standard in format and arithmetic. By contrast, low-precision integers (for example, `int8`) are not governed by IEEE 754: their format and arithmetic can be implementation-dependent. We assume a signed two's complement format in the publications forming this thesis due to its popularity.

[7] Concretely, STE applies $\mathcal{Q}$ in the forward pass but, during backpropagation, replaces $\frac{\partial \mathcal{Q}}{\partial x}$ by 1, that is, it passes gradients through as if $\mathcal{Q}$ were the identity.

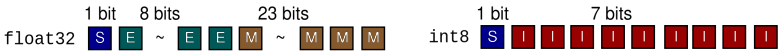


Figure 2.4.: Conceptual visualization of `float32` and `int8` formats. For `float32`, the binary representation consists of a sign bit (S), an 8-bit exponent (E), and a 23-bit mantissa (M), where the value is computed as $(-1)^S \times 1.M \times 2^{E-127}$. In contrast, `int8` uses an 8-bit signed integer representation, where the most significant bit encodes the sign and the remaining bits represent magnitude. This distinction leads to vastly different bit-level sensitivities in perturbation-based attacks.

can inflate the value by as much as 3.4×10^{38}, leading to extreme neuron activations that dominate the network [HFK+19, LWLX17].

2.3.3. Gradient-Based Bit-Search

The *Progressive Bit-Flip Attack* (PBFA), introduced in the seminal work by [RHF19], forms the methodological fundament of BFAs employing gradient-based bit-search techniques, as most other, more specialized (for example, targeted) BFAs designed to degrade CNNs have been derived from PBFA.

PBFA targets a quantized and trained CNN Φ and employs a technique called *Progressive Bit Search* (PBS) to choose bits for flipping. Let $\mathbf{W}_q^{(l)}$ denote the quantized weight tensor of layer l. PBS begins with a forward and a partial backward pass, performing error backpropagation without updating weights on a randomly chosen batch $\mathbf{X}$ of training data with a target vector $\mathbf{t}$. Next, for each layer l, PBS selects the weights corresponding to the top-k largest binary encoded gradients as potential candidates for bit flipping. For each candidate bit, PBS then constructs a perturbed weight tensor $\widehat{\mathbf{W}}_q^{(l)}$ by flipping exactly this bit in $\mathbf{W}_q^{(l)}$. Among all such one bit perturbations in the set of candidate bits, PBS finds the bit that maximizes the difference between the loss $\mathcal{L}$ of the perturbed CNN and the loss of the unperturbed CNN,

$$\max_{\{\widehat{\mathbf{W}}_q^{(l)}\}} \mathcal{L}\Big(\Phi(\mathbf{X};\ \{\widehat{\mathbf{W}}_q^{(l)}\}_{l=1}^{L},\ \mathbf{t}\Big) - \mathcal{L}\Big(\Phi(\mathbf{X};\ \{\mathbf{W}_q^{(l)}\}_{l=1}^{L},\ \mathbf{t}\Big), \tag{2.2}$$

via iterative testing, whereby the same $\mathcal{L}$ is used that was minimized during training, for example, (binary) cross entropy (B)CE for (binary) classification. The process is repeated until a predefined number of bit flips is reached, whereby later repetitions include previously flipped bits during the initial forward and a partial backward pass as well as construction of the perturbed weight tensors $\widehat{\mathbf{W}}_q^{(l)}$.

Subsequently, we refer to this specific attack as PBFA and use the term BFA to refer to a broader range of attacks that systematically induce malicious bit flips in a neural network's weights and biases.

2.3.4. Defenses for CNNs

Various methods have been proposed to defend CNNs against BFAs. Network hardening methods, such as adversarial [HRL+20, LRX+20] and perturbation-resilient training [CMDLP23], as well as gradient obfuscation strategies [ZWC+22, WZW+23], aim to offset gradient-based bit search attacks either via learning weights that render the model inherently perturbation resilient or via diverting the attacker's gradient-based bit search algorithm towards relatively insensitive parameters of the model. As setting up such defenses typically requires (re)training, their ex-post deployment capabilities are limited, and they do not necessarily correct compromised neurons. Detection-oriented methods rely on efficient hashing algorithms [JK21, LRH+21] or output code matching [OL22] to identify bit flips effectively, achieving high detection rates in the computer vision domain. These techniques are often compatible with existing pre-trained networks and can be applied post model deployment. Their restoration capabilities, however, remain limited, and they typically require model retraining or pruning of the compromised weights. In contrast, proactive defenses seek to anticipate attacks and associated vulnerable weights through analyzing assumed underlying attack mechanisms [LYW+23, LRX+20], such as the tendency of the gradient-based bit-search employed by most BFAs to yield attack solutions involving bits near the most significant bit[8] (MSB) or the MSB itself of weights with large gradients. These approaches prioritize weight restoration over pruning and either aim to revert the network to its pre-attack state [LYW+23] or use statistical approximation to reduce the impact of compromised weights [LRX+20].

2.3.5. BFAs and GNNs

In principle, PBFA and, thanks to the flexibility of the gradient-based bit-search algorithm PBS, potentially most other BFA variants based on it can be directly extended to GNNs. However, [HMDD22] demonstrate that the high susceptibility of CNNs to BFA is closely linked to the weight sharing characteristic of convolutional filters. Moreover, [HMDD22] show that MLPs are inherently more robust to PBFA than CNNs due to their different gradient distributions. Despite the absence of convolutional filters in GNNs, as well as MLPs being an integral component of certain GNNs such as GIN, no specialized attacks have been developed so far. This gap is also reflected in the wider body of research.

Existing research on the resilience of GNNs to bit flips focuses exclusively on floating-point representation and did not produce any adaptations specifically tailored to GNNs: [JWL+22] study random bit faults but do not consider expressivity and [WYW+24] consider a variant of PBFA modified to flip bits in the exponent of a weight's floating-point representation but exclusively focus on output verification (that is, verifying that the

[8] In IEEE 754 `float32`, the most significant bit is the sign; it is followed by 8 exponent bits and 23 fraction (mantissa) bits, see Figure 2.4. Flipping the sign bit flips the sign; changing an exponent bit shifts the scale; changing the least significant fraction bit adjusts the value by the smallest step at that scale. In two's-complement `int8`, the most significant bit is also the sign and the remaining bits encode the magnitude; flipping the least significant bit changes the value by one, while flipping the most significant bit changes the sign. Integer overflow typically wraps around unless saturating (clamping) arithmetic is used.

output stems from an unperturbed model). Specifically, [WYW+24] probe an untrusted GNN by generating predictions for a small set of carefully chosen fingerprint nodes and checking whether these predictions match what would be expected as output by a clean model.

2.4. Lottery Ticket Hypothesis

Related work on LTH for NNs and CNNs largely falls into two main categories: one focused on theoretical and empirical insights into the mechanisms of sparse neural network training, and the other centered on translating these insights into practical applications. A subset of works bridges both areas, offering contributions that are both conceptually and practically important.

The first work describing LTH falls into the last category, as it postulates LTH, empirically tests it, and proposes an iterative pruning scheme for generating winning lottery tickets [FC18]. [FDRC19] build upon LTH by proposing Iterative Magnitude Pruning with rewinding, which prunes subnetworks early in training instead of at initialization, enabling the discovery of sparse winning tickets in deep networks such as ResNet-50 [HZRS16] while preserving prediction quality and training stability. By showing over-parameterized networks contain subnetworks that achieve high prediction quality without training, with theoretical guarantees for deep and shallow networks.

Focused on building the theoretical framework around LTH, [MYSSS20] prove aforementioned SLTH (see Section 1) as an extension of LTH and [dCNV22] prove SLTH for convolutional networks, showing that even without training, large randomly initialized CNNs can be pruned to subnetworks that closely approximate trained models. [ZWL+21] provide a theoretical explanation for the improved generalization observed in LTH, showing that pruning enlarges the convex region of the optimization landscape, which facilitates faster convergence and requires fewer samples to achieve zero generalization error. [ZJZ+21] further support LTH using Inertial Manifold Theory, showing that subnetworks obtained via pruning can achieve comparable performance to dense models without repeated pruning and retraining.

Concentrating on the practical implications of LTH, [ZCSL19] introduce Eager Pruning, an LTH-motivated approach that prunes deep neural networks early during training, reducing computational cost without sacrificing prediction quality. Recognizing winning tickets as valuable intellectual property, [CCZW21] formulate a lottery verification problem and propose to treat the binary sparsity mask of a winning ticket as a structural carrier for ownership information.

Works on LTH for GNNs are likewise largely practically motivated, aiming to increase computational as well as memory efficiency. These works typically treat pruning the graph and the GNN's trainable parameters as a unified task. Specifically, [TP23] formulate the Graph Lottery Ticket (GLT) Hypothesis, suggesting that every graph contains a sparse substructure that can support the prediction quality of graph learning models and [CSC+21] propose the Unified GNN Sparsification (UGS) framework, which simultaneously prunes adjacency matrices and the model's parameters to uncover winning GLTs, which

was later enhanced further by [HYMK23]. Based on these ideas, a substantial number of such co-pruning frameworks [WLW+22, SWC+23, ZWH+24] were created, whereby some of them drew additional inspiration by the aforementioned theoretical works LTH and SLTH [YXJ+24, YIGA+24]. The common property of these works is that they offer few if any theoretical explanations and do not consider expressivity in its formal sense, namely the degree to which a GNN is capable of distinguishing non-isomorphic graphs.

3. Contributions

In this chapter, we explain the methodology used in each of the papers that together form this thesis in brief, discuss their contribution and common properties.

We present the first BFA dedicated to GNNs, specifically seeking to degrade the network's ability to differentiate between non-isomorphic structures (Paper 1). Subsequently, we propose an associated defense mechanism that is specifically designed to detect a BFA and verifiably reset the target model to its pre-attack state (Paper 2). Moreover, we provide the first theoretical study of how BFAs affect GNN expressivity, and in that context, show the existence of upper bounds to the number of bit flips a GNN's expressivity can withstand and as well identify certain particularly vulnerable settings (Paper 3). Extending our research from quantization as a means of gaining efficiency, we provide the first theoretical connection between winning lottery tickets and expressivity, prove a novel, GNN-specific extension of the classical SLTH that considers expressivity and analyze its relation to training dynamics (Paper 4).

3.1. Shared Concepts Among Our Contributed Research

The common ground of Paper 1 through Paper 3 is their focus on BFAs. Paper 1 and Paper 2 are specifically related through their focus on the robustness of integer quantized GNNs to BFAs, whereby Paper 2 specifically seeks to address the vulnerabilities created by Paper 1; addressing certain deficits of existing BFA defenses ported from CNNs along the way. Paper 3, while likewise being focused on BFAs, deviates from the use of integer quantized GNNs that were the center of our attention in Paper 1 and Paper 2 and instead focuses on models using floating-point representations in order to allow us to investigate the effects of the more diverse binary semantics (see Figure 2.4) on robustness. Paper 1 and Paper 3, though, share exploration of the relation between expressivity and robustness to BFAs as their common denominator. Paper 4 digresses from the focus on BFAs but, through its investigation of the relation between LTH and expressivity in GNNs, shares a common ground with the other three works in that it investigates a relationship along the efficiency-expressivity axis. In doing so, Paper 4 lays the groundwork for interlinking robustness and efficiency via sparsification through the lens of expressivity in the future, pointing toward the study of robust winning lottery tickets (see Section 4.3) and linking back to the thesis's focus on BFAs (Paper 1 through Paper 3). A high-level visualization of these commonalities is provided in Figure 1.1 (Section 1.1).

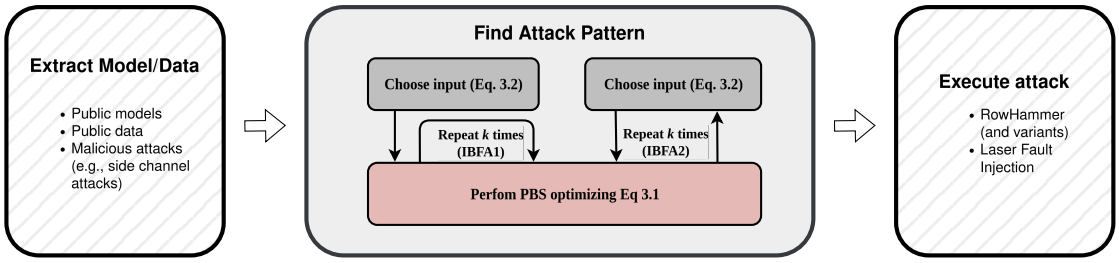


Figure 3.1.: Conceptual visualization of IBFA's (represented by the central panel) integration with methods for obtaining model parameters and data, as well as with practical approaches for inducing bit flips in hardware discussed in Section 2.3.1. In both variants, IBFA leverages PBS to optimize Equation (3.2) for samples selected according to Equation (3.1), in order to identify destructive bit-flip patterns. Note that while the figure above depicts bit search and the realization of bit flips in hardware as two separate, sequential tasks, it is also possible to treat them as a single task and directly execute an attack on each vulnerable bit identified.

3.2. Attacking GNNs with Bit Flips

In Paper 1, we introduce the first BFA specifically dedicated to GNNs. Whereas, in principle, PBFA and potentially most other BFA variants based on it (see Section 2.3) can be directly ported to GNNs due to the versatility of the underlying gradient-based bit-search, none of them considers expressivity as a primary target. Instead, most approaches operate by conducting PBS with respect to some loss function that is reliant on the classification labels. We present a first study evaluating the robustness of integer quantized GNNs under PBFA and random bit flips, revealing that their effectiveness remains limited in settings where distinguishing graph structures is critical, showing only marginal improvement over random bit flips. To illustrate that such GNNs can still be compromised in the given setting, we introduce the Injectivity Bit-Flip Attack (IBFA), a method designed to impair the expressivity of neighborhood aggregation mechanisms in GNNs. Similar to PBFA, IBFA assumes a gray-box attack model in which the attacker knows the target model's parameters and has access to a small unlabeled data sample. We concentrate on the GIN model, which achieves 1-WL expressivity through injective aggregation functions for suitable parameter choices[1]. GIN is widely deployed in practice and integrated into standard libraries [FL19, BF23, WLL+23, GHYS22, YJLH22]. Conceptually, IBFA inverts PBFA's objective: rather than increasing the training loss (Section 2.3.3, Equation (2.2)), it aligns the model's outputs for inputs the clean network deems maximally different.

[1]See Section 2.2.1 for the connection to the hashing approach function used in 1-WL.

Optimization Objective and Sample Selection. To operationalize this attack, we select data points (or batches) $\mathbf{X}_a$, $\mathbf{X}_b$ according to

$$\arg\max_{\{\mathbf{X}_a,\mathbf{X}_b\}} \mathcal{L}\Big(\Phi(\mathbf{X}_a;\{\mathbf{W}_q^{(l)}\}_{l=1}^L),\ \Phi(\mathbf{X}_b;\{\mathbf{W}_q^{(l)}\}_{l=1}^L)\Big) \tag{3.1}$$

and depart from PBFA's loss-maximization objective and instead employ PBS to optimize

$$\min_{\{\widehat{\mathbf{W}}_q^{(l)}\}} \mathcal{L}\Big(\Phi(\mathbf{X}_a;\{\widehat{\mathbf{W}}_q^{(l)}\}_{l=1}^L),\ \Phi(\mathbf{X}_b;\{\widehat{\mathbf{W}}_q^{(l)}\}_{l=1}^L)\Big) \tag{3.2}$$

for either L_1 or Kullback-Leibler-Divergence (KL) [KL51] as $\mathcal{L}$, depending on the type of classification task[2].That is, we seek to minimize the difference in the predictions of the perturbed GNN for the two graphs or batches of graphs that receive the most distinct output in the unperturbed GNN. We provide two variants of IBFA; one where the input is selected once via Equation 3.1 and remains unchanged during subsequent PBS iterations and one where input selection is executed after every attack iteration via substituting $\widehat{\mathbf{W}}_q^{(l)}$ for $\mathbf{W}_q^{(l)}$ in Equation 3.1, addressing the concern that optimal inputs could change after a bit flip.

Intuitively, IBFA seeks to flip bits in the GNN's trainable parameters such that, at every iteration, the outputs for the two graphs or batches of graphs that have the most distinct predictions (that is, softmax outputs) are aligned with one another. Under the assumption that the learned predictions correlate with different (that is, 1-WL distinguishable) graph structures, repeated application of this algorithm will necessarily break the GNNs abilities to distinguish those and consequentially impact the learned injectivity of the aggregation functions. A high-level integration of IBFA with methods for obtaining model parameters and data, as well as with practical approaches for inducing bit flips in hardware (see Section 2.3.1), is shown in Figure 3.1.

Empirical Evaluation. The empirical evaluation, which focused on GIN and involved 10 typical yet diverse benchmark datasets, revealed that IBFA typically induced much greater damage than random bit flips or PBFA, particularly also in those settings where PBFA failed to induce significantly more degradation than the random bit-flip attack. Moreover, our results were consistent with the hypothesis that, particularly in datasets where there is a strong connection between class membership and structural graph similarity, IBFA yields more degradation than PBFA. Furthermore, IBFA achieves this with an average of 33 bit flips, which falls within the plausible range an attacker can realize according to prior work (24-50; [WZW+23, YRF20]). The different attack patterns exhibited by IBFA and PBFA were examined as well, and we found that IBFA displays a much higher

[2]For binary classification tasks, where the GNN outputs are $n \times 1$ vectors of Bernoulli probability mass functions (PMFs), we use the L_1 loss due to its simplicity and sufficiency to optimize Equation (3.2). For multiclass or multitask binary classification, where outputs are $n \times m$ matrices of categorical PMFs, L_1 fails to capture class-wise differences effectively. In these cases, we instead use the pointwise discrete KL to compare per-sample distributions, allowing the attack to align the output distributions more effectively.

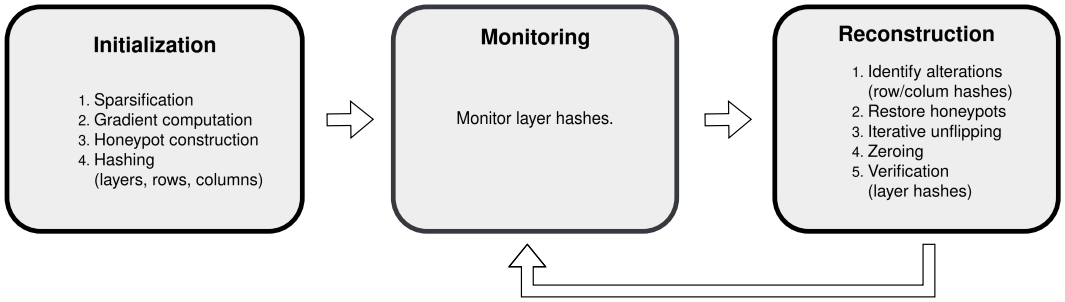


Figure 3.2.: Conceptual overview of Crossfire's three-stage defense pipeline. In the initialization stage, the GNN is sparsified, honeypots are embedded based on gradient information, and integrity hashes are computed at the layer, row, and column levels. During monitoring, Crossfire verifies model integrity through layer-level hashes, enabling fast perturbation detection. In the reconstruction stage, perturbations are localized via fine-grained hashes and resolved by restoring honeypots, iterative bit-level unflipping, or zeroing. Final verification ensures that the reconstructed model retains its integrity.

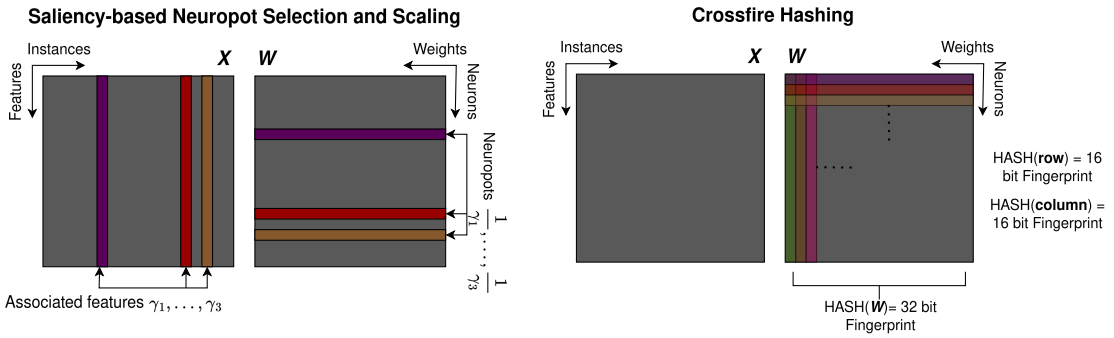
diversity of which layers are selected for a bit flip, whereas PBFA tends to flip bits in the input layer; IBFA spreads its bit flips across the entire network.

3.3. Defending GNNs From Bit Flips

Paper 1 revealed that GNNs in settings that are otherwise highly resilient to BFAs, such as PBFA, can be degraded nonetheless by attacks tailored to their specific mathematical properties. Hence, in a natural progression of our research activities, in Paper 2, we studied two representative BFA defense methods that were initially developed for CNNs and assessed their effectiveness for GNNs under attack by IBFA and PBFA.

Baseline BFA Defenses: NeuroPots and RADAR. The first approach was of proactive nature: NeuroPots [LYW+23] embeds honeypots as vulnerabilities to lure attackers employing gradient-based bit-search and facilitate fault detection and model restoration. These honeypots are created via downscaling a certain percentage of weights in each trainable parameter matrix by a global factor γ (which is a hyperparameter chosen by the defender) and upscaling the corresponding inputs with the reciprocal factor such that the gradients for these weights are artificially enlarged[3] . These operations are applied such that they do not affect the network output, however, except for rounding errors if the network is quantized. This random selection of honeypot neurons may miss critical units or include inactive ones, though, and the use of a global scaling factor γ neglects

[3]Recall that PBS ranks bits based on the magnitude of the loss gradients with respect to their underlying weights; see Section 2.3.3. Hence, any PBS-based BFA can be steered toward specific weights through artificially enlarging their gradients. Hashing-based detection can then focus on those weights, and, if perturbed, they can be cheaply restored from a securely stored copy.



(a) Conceptual visualization of honeypot construction within a layer's weight matrix. Different scaling factors γ_i are applied to selected rows of the matrix designated as honeypots. The values of γ_i, used to rescale both the columns of the input $\mathbf{X}$ and the corresponding rows of the weight matrix $\mathbf{W}$, are determined based on gradient information.

(b) Conceptual visualization of Crossfire hashing applied to the weight matrix $\mathbf{W}$. The 16-bit row and column hashes enable accurate localization of weight perturbations caused by bit flips in most practical cases. The method itself remains agnostic to the specific choice of hash function.

Figure 3.3.: Conceptual visualizations of the two core components of Crossfire. (a) Honeypot construction within a layer's weight matrix, where scaling factors γ_i are applied to selected rows based on gradient information. (b) Crossfire hashing of the weight matrix $\mathbf{W}$, where 16-bit row and column hashes enable accurate localization of weight perturbations caused by bit flips.

neuron-specific functional roles. More importantly, the method cannot guarantee full restoration to the pre-attack state, necessitating post-hoc evaluation since it does not detect perturbations in non-honeypot weights.

The second approach (RADAR) [LRH+21] was detection-focused. Within each layer, RADAR organizes weights into groups and uses a checksum-based algorithm to generate a 2-bit to 3-bit signature per group. To detect BFAs at runtime, this signature is computed and compared with a stored reference value. In case any modification in a group is detected, the weights in the affected group are set to zero to minimize the impact on prediction quality. While RADAR demonstrates high detection rates and is not dependent on the assumption that an attack employs gradient-based bit-search, its restoration strategy of zeroing entire weight groups only yields a partial recovery of the model's prediction quality. Moreover, RADAR is unable to guarantee a return to the pre-attack configuration by design, necessitating post-recovery validation. For very large numbers of perturbed weights, the extensive zeroing of entire groups may introduce collateral damage and degrade the network.

Design of the Crossfire Defense. Based on these observations, we designed Crossfire with the intent to not only leverage both RADAR's and NeurPots' conceptual strengths without

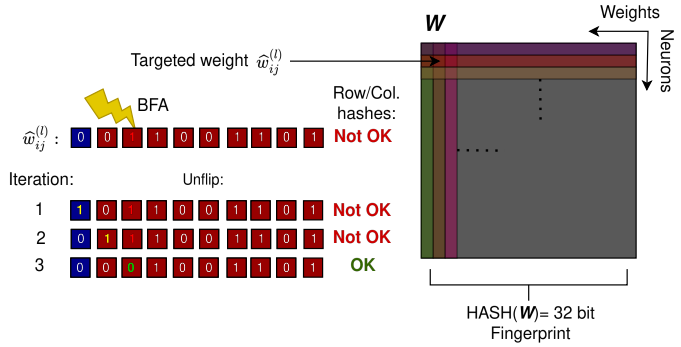


Figure 3.4.: Conceptual visualization of the Crossfire recovery procedure. The defense first detects attacks by checking 32-bit weight matrix hashes. If an attack is detected, 16-bit cross-hashes are then evaluated to localize weight changes. When a honeypot weight is affected, it is directly replaced; if a non-honeypot weight is hit, iterative unflipping is performed (see above). If unflipping is unsuccessful, the weight is set to zero as a final fallback, leveraging the pre-deployment–induced sparsity. Finally, verification is conducted using 32-bit matrix hashes to ensure restoration to the model's pre-attack state.

inheriting their weaknesses but also to go beyond and seek full, verifiable reconstruction. Crossfire operates in three sequential stages: initialization, monitoring, and reconstruction. In the initialization phase, the fully trained and quantized GNN is sparsified through dequantization and L_1 pruning. Honeypots are then strategically embedded based on gradient information and network depth (Figure 3.3a), and multi-level hashes, computed at the layer, row, and column levels, are established to enable integrity checks (Figure 3.3b). During monitoring, Crossfire continuously verifies the integrity of the model via layer-level hashes, enabling the fast detection of perturbations. Upon detection, the reconstruction phase is triggered, where perturbed weights are localized using fine-grained row and column hashes. Depending on the nature and location of the perturbation, weights are either restored using securely stored[4] honeypot backups, corrected at the bit level if the perturbation is out-of-distribution, or zeroed if neither restoration path succeeds (Figure 3.4). Importantly, the induced sparsity ensures that zeroing is less harmful and more likely to revert the network to its original behavior. After reconstruction, the integrity of the model is verified via the layer hashes generated at initialization. A high-level overview integrating the three stages of the Crossfire defense pipeline is shown in Figure 3.2.

While Crossfire works best under a gray-box threat model similar to that described in Section 3.2, it conceptually also mitigates NeuroPots' limitations in a (relaxed) white-box setting: even if an attacker knows, for example, which weights are honeypots, Crossfire's redundant architecture (hash-based detection, bitwise unflipping, and zeroing) can, in

[4]Similar to NeuroPots, we assume that such a secure storage can be realized via trusted execution environments (TEE) like Intel SGX [MAA+16] or ARM Trustzone [PS19, ARM09].

principle, still detect perturbations and restore the network.

Empirical Evaluation. The empirical results, which are based on a similar setup as described in Section 3.2, for up to 55 bit flips (whereby 50 is considered an upper boundary to what an attacker can practically realize [WZW+23]) demonstrate that Crossfire consistently outperforms NeuroPots and RADAR across multiple representative GNN architectures and datasets. Crossfire achieves near-complete recovery of predictive performance even when the victim GNN is attacked by strong IBFAs and is capable of verifiably resetting the model to its pre-attack state significantly more often than its competitors. Computationally, we find that Crossfire's overhead is one order of magnitude smaller than even the simplest MP layer, and its storage overhead scales effectively with increasing weight matrix sizes.

3.4. The Relationship Between Robustness and Expressivity of GNNs

In Paper 3, we take on a more theoretical perspective. Motivated by [AGA+23]'s results, which indicate that GNN expressivity is generally resilient to random bit flips, we derive upper bounds to the number of malicious bit flips the expressivity of a GNN can resist and explore how these bounds increase or decrease depending on parameter count, number of bits used to represent a value (bit width), diversity of distinguishable subgraphs, variety of GNN embeddings, input feature encoding, dataset homophily[5], activation function, and the bit's location in the numerical representation. Empirically we confirm our hypotheses regarding those impact factors using GNNs represented in floating-point representation, which allows us to additionally study how the semantics (see Section 2.3.2) underlying the flipped bits affect the GNNs vulnerability.

Robustness Bounds. To that purpose, we first reformulate the seminal result that, for a message-passing GNN to be equally expressive as the 1-WL test, it is sufficient that all aggregate and combine functions are injective [XHLJ19]. Specifically, and based on previous results on global injectivity of Rectified Linear Unit (ReLU) activated MLPs [PKL+22], we reformulate the condition via the dot product such that in every trainable parameter matrix of every MLP in the GNN, there must be at least one parameter vector that leads to a distinguishable dot product for any two distinct input vectors on a bounded dataset.

Using this condition, we characterize the robustness of GNN expressivity by analyzing how bit-level perturbations affect these dot products and their ability to yield distinguishable node and graph (intermediate) representations, as follows. Assuming an arbitrary

[5]Homophily is the tendency of adjacent nodes to share attributes or labels [JWG+22, ZLC24]. High homophily reduces the diversity of neighborhood multisets (aggregates), which can exacerbate (over)smoothing [YHS+22] and in turn ease classification [Ker22] in certain settings. In our setting, attribute homophily tightens the derived bounds.

but injective activation and a neural-moment (see [AGA+23]) architecture representative of the class of MP-GNNs, we find that the upper bound to break the guaranteed distinguishability at the node level at a given MLP layer is, besides the obvious factor of bit-width and matrix dimensions, dependent on the number of distinct elements of the (intermediate) representations of the two most distinct node embeddings in that MLP's input domain. While this, without constraints on graph structure, input features, or the attacked layer (which we discuss later), allows for the construction of a wide range of vulnerability scenarios, it sets the foundation for the more interesting graph level upper bound.

The graph level upper bound is derived using the fact that the representative MP-GNN architecture we analyzed will, by construction, at a given layer at most yield as many distinct node embeddings as the 1-WL algorithm would for the corresponding number of iterations. Hence, we find that to render two graphs indistinguishable at a given layer, the aforementioned node level bound scales with the number of nodes of the two graphs that receive distinct colors by the 1-WL algorithm for a number of iterations corresponding to the layer's depth. This is a notable finding, as it shows that the susceptibility of graph level expressivity to BFAs is directly related to the diversity of distinguishable subgraphs present in the two graphs, which corresponds to the number of different colors output by the 1-WL.

Mitigating and Aggravating Factors Concerning mitigating and aggravating influence factors to above bounds, we first discuss activation functions. We show that GNNs employing ReLU-activated MLPs are particularly vulnerable: in signed representations, flipping the sign bits of the targeted weights is sufficient, so that ReLU sets the resulting negative pre-activations to zero and the graph- and node-level bounds no longer depend on the representation's bit-width. Moreover, we show that under the assumptions of one-hot[6] encoded node features of the input graphs (a common practice with typical benchmark datasets, such as [MKB+20]), BFAs on the first layer can also be particularly devastating. In this setting, the maximum node degree across the two graphs determines how many different non-zero aggregated node features appear at the MLP input, which in turn reduces the number of weights an attacker must perturb to degrade the guaranteed expressivity. We further show that a high degree of homophily is an aggravating factor in such a setting, leading to further reduced bounds, as the the variety of node aggregates tends to be lower on highly homophilous graphs.

Empirical Evaluation. Empirically, we test the described mitigating and aggravating factors as follows. For representative GNN architectures (GIN and GCN) and benchmark datasets, we conduct random initialization and subsequently induce bit flips from 1 to 0 in mantissa and exponent bits and from 0 to 1 in sign bits at randomly chosen locations in

[6] In one-hot encoding, a categorical label $c \in \{1, \ldots, m\}$ is represented by the m-dimensional standard basis vector $e_c \in \{0, 1\}^m$ with a single 1 at index c and 0 elsewhere. This yields mutually orthogonal, equidistant feature vectors that make no ordinal assumptions about categories, at the cost of increased dimensionality. For graphs, node types (for example, atom species) are commonly encoded this way.

the GNNs parameter matrix[7]. Over a series of more than 100000 individual experiments, we were able to confirm the above-mentioned influence factors impact on the number of bit flips required to annihilate GNN expressivity. Specifically, we found that ReLU-activated GNNs operating on highly homophilious graphs with low-dimensional or one-hot encoded features are a particularly vulnerable configuration.

3.5. Why Expressive Lottery Tickets Win

In Paper 4, we took our investigation of the interplay between expressivity and efficiency beyond quantization as a means to reduce computational and storage complexity and took a theoretical perspective analyzing the connection between the expressivity of a sparsely initialized GNN and its chance of being a winning lottery ticket (that is, yielding post-training prediction quality comparable to that of the dense model; see Section 1). Previous work on LTH and GNNs was mainly method-driven and provided few to any theoretical results and no connection between expressivity and LTH explicitly. Therefore, we were motivated to establish such a link as a stepping stone for future work (see Section 4.3) on the robustness of winning lottery tickets (that is, investigating the research question "Do winning lottery tickets exist that are more robust to BFAs than others?"). This would provide a sparsification-based perspective on the relationship between efficiency, expressivity, and robustness, thereby complementing our quantization-focused study in Paper 1 and building on our earlier results from Paper 3, which linked expressivity and robustness to malicious bit flips. See Section 4.3 (Future Work) for details.

A General Expressivity Criterion and Computational Graph View. To this purpose, we first provide a general criterion that any pruning mechanism, whether it is pruning the input graphs or the models trainable parameters, must satisfy to preserve the theoretical ability to reach maximal prediction quality on a given dataset. Specifically, we establish that any modifications done to a GNN (regardless of whether it is pruning, quantization, or architectural adaptations) or any modifications of the graph dataset (regardless of whether nodes or edges are pruned or features are transformed) must maintain the GNNs theoretical ability to distinguish at least those non-isomorphic graphs that belong to different classes of the dataset.

LTH is concerned with a setting of pre-training pruning, where the assumption is that after a sparse initialization, only the non-zero weights are updated. Hence, by using a computational graph representation for each MLP in the GNN, where nodes represent neurons, edges their connection, and learnable parameters edge weights, we separate the initialization-time pruning from concrete numerical values of the weights, which can change to potentially arbitrary values during training.

[7]We chose these semi-randomly induction of bit flips in our experiments as previous results [AGA+23] indicate that GNNs are maximally expressive for completely randomly drawn weights in almost all cases.

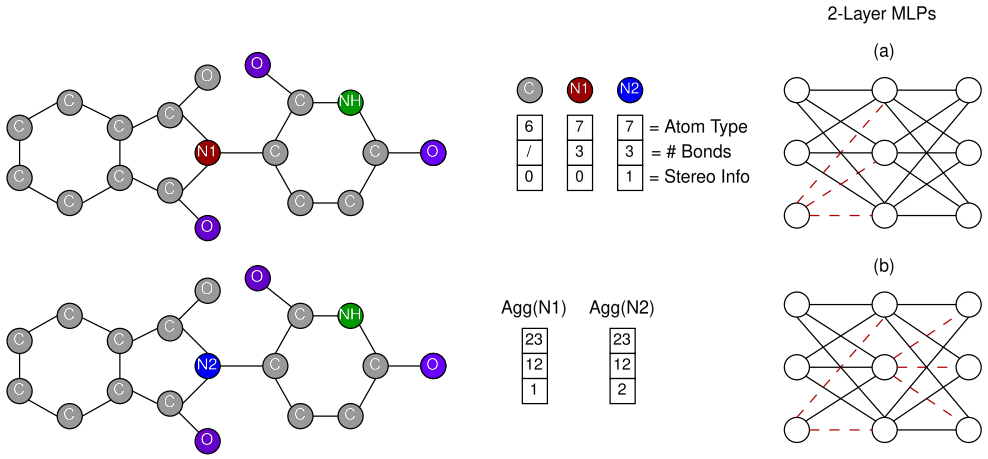


Figure 3.5.: Visualization of thalidomide (also known under its trade name Contergan) and its embryotoxic stereoisomer [LK62, BÏ14] as an example for which a failure to distinguish the two can have potentially life-altering consequences for patients (left-hand side of the figure). As shown in the center of the figure, the aggregates of nodes N_1 and N_2 in the first layer differ by only a single feature. A pruning mask removing the red dashed edges from the exemplary MLPs (a) and (b) applied to these aggregates would irrecoverably cancel this difference; hence these edges are part of critical path sets of a GNN trained to predict different classes for the two molecules, as in, for example, toxicity categorization tasks.

The Strong Expressive Lottery Ticket Hypothesis. Using this representation, we then derive a formal definition of critical computational paths[8] in the MLPs of a GNN, stating that a set of computational paths is critical if, for the computational graph's remaining edges , no weights exist such that the pruned GNN can reach a prediction quality at least as good as its corresponding dense counterpart. There is a direct connection to our aforementioned criterion, in the sense that any initialization-time parameter pruning of the GNN must at least leave those computational paths intact for which weights exist that allow the distinction of at least those non-isomorphic graphs that are associated with different classes.

For sufficiently overparameterized GNNs, we then proceed to formally show the existence of such paths and corresponding weights that yield 1-WL level expressivity, as well as the trainability of these weights. This theoretical result constitutes a novel variant of SLTH, to which we refer to as the Strong Expressive Lottery Ticket Hypothesis, or SELTH.

Moreover, we formally show that expressive sparse initializations can improve generaliz-

[8] By a *computational path* in an MLP we mean a sequence of neuron-to-neuron connections that carries a signal from an input neuron to an output neuron across consecutive layers. Pruning applies a binary mask to the edges of the graph modeling the MLP; removing an edge eliminates all paths that traverse it, preventing their contributions from reaching the output in the forward pass.

ation and convergence, using Gradient Diversity[9] as a proxy [YPL+18]. Furthermore, we identify cases where expressivity loss is irrecoverable, exemplified by molecular property prediction in a medical context, which is illustrated in Figure 3.5.

Empirical Evaluation. Empirically, we find that sparse initializations which remain expressive at the start of training are far more likely to become winning tickets, that is, they reach the same (or better) post-training accuracy as the dense model with fewer parameters. In contrast, sparsity patterns that already collapse distinct structures at initialization rarely regain that discriminative ability during training. This trend holds across multiple benchmark datasets, pruning levels, and standard GNN backbones (that is, GIN and GCN), and it aligns with the aforementioned theoretical considerations.

[9] Gradient Diversity informally captures how varied the learning signals are across samples. When gradients point in similar directions, updates are redundant and learning progress slows; when they differ in informative ways, the optimizer receives a broader, more useful signal.

4. Conclusion

This chapter discusses our results and relates them to the research goals in Section 1.1, provides an overview of assumptions and limitations, offers an outlook on future work, and concludes with final remarks.

4.1. Discussion

BFAs and Expressivity With IBFA (Paper 1), we propose the first BFA that specifically targets the expressivity of GNNs. By leveraging their ability to distinguish non-isomorphic graph structures, IBFA exposes a critical vulnerability and achieves degradation even in settings where standard attacks like PBFA show limited effect. This first work shows the importance of properly protecting GNNs from BFAs, as quantization alone may not offer effective protection, and underlines the importance of conducting future research in that direction.

The importance of expressivity is further emphasized by our theoretical framework for understanding the resilience of neural moment-based GNNs to BFAs, which yielded provable bounds on the number of bit flips needed to impair expressivity (Paper 3). Our analysis reveals that this robustness of GNN expressivity is fundamentally shaped by architectural design, the structural characteristics of the input graphs, the choice of activation functions and has potential implications to practitioners. For example, ReLU-activated GNNs exhibit increased vulnerability to bit flips, especially when considering limited feature dimensionality and one-hot encoded node features.

In light of this, Paper 1 and Paper 3 provide answers to the first and third research goals discussed in Section 1.1, as they show that the CNN-derived intuition about the robustness of integer-quantized models does not carry over to GNNs once attacks exploit GNN-specific sources of expressivity and the resilience of GNN expressivity to malicious bit flips admits formal characterization via dataset- and architecture-dependent bounds, identifying more and less robust configurations (for example, levels of homophily, feature encodings, and activation choices). We find the results on homophily are particularly noteworthy, as similar observations have been reported for poisoning and evasion attacks [ZJL+22]. Although orthogonal to BFAs, these parallels motivate future investigations into how evasion and poisoning relate to BFAs.

Increasing GNN Robustness to BFAs. Crossfire (Paper 2) merges proactive and detection-based concepts with an efficient verification strategy into a cohesive pipeline, overcoming key limitations of methods transferred from CNNs, such as the lack of full

restoration guarantees in NeuroPots and the coarse-grained recovery of RADAR. It highlights the importance of a multilayered defense framework with redundant detection and reconstruction stages and built-in ex post verification of the reconstruction process.

Furthermore, as revealed in Paper 3, to improve robustness, practitioners may consider using Sigmoid Linear Unit (SiLU) activations instead of ReLU as a more resilient alternative or enhance feature dimensionality through pre-coloring strategies such as those proposed by [AGKG22, CW19]. Moreover, converting one-hot encoded features into dense representations could help preserve expressivity while mitigating susceptibility to BFAs targeting expressivity.

In summary, Paper 2 and Paper 3 deliver on the second and third research goals outlined in Section 1.1: Crossfire contributes a multilayered defense that combines proactive honeypots, precise detection, and efficient post-attack verification to enable verifiable restoration, overcoming the lack of full integrity guarantees and coarse recovery in prior CNN-transferred methods, and the theory in Paper 3 indicates actionable robustness levers (for example, preferring SiLU over ReLU, increasing feature dimensionality via pre-coloring, and replacing one-hot with dense encodings).

Winning Lottery Tickets. We establish the first formal link between the LTH and GNN expressivity (Paper 4). By introducing and proving the SELTH, we demonstrate that sparse subnetworks can retain the expressive power of 1-WL and that preserving expressivity in sparse initializations is critical for identifying winning tickets. Our analysis shows that expressive sparse initializations not only safeguard against irrecoverable loss of discriminative power but also potentially accelerate convergence and improve generalization. The work highlights the risks of pruning strategies that discard critical paths, which can permanently impair expressivity in practical domains such as drug discovery, and points to future directions in developing pruning mechanisms and sparse initialization strategies that explicitly preserve expressive capacity.

Overall, Paper 4 delivers on the fourth research goal outlined in Section 1.1: it establishes the missing theoretical link between LTH and GNN expressivity via SELTH, identifies conditions under which sparse initializations preserve 1-WL expressive power, and translates these insights into guidance for expressivity-preserving pruning and initialization, thereby laying the groundwork for efficient graph learning models that maintain expressive capacity.

4.2. Limitations

Threat Models. Both IBFA and Crossfire (Paper 1 and Paper 2) adopt a white-box threat model, assuming that an adversary can precisely manipulate individual bits in quantized GNN weights during inference through mechanisms such as RowHammer, without budgetary restrictions on the number of flips. While this worst-case framing is standard in prior BFA literature on CNNs, it arguably overstates the practical capabilities of real-world adversaries, for whom reliably inducing more than a few dozen targeted flips remains technically challenging. Moreover, the assumed attacker knowledge (including

access to the network architecture, quantization scheme, and in some cases subsets of training data) represents a best-case scenario for the adversary that may not always hold in practice. These assumptions were necessary for researching a first attack and evaluating the vulnerability of GNNs and for benchmarking defense mechanisms under the strongest conceivable conditions. However, they limit the immediate applicability of the results to scenarios where attackers operate under more constrained, noisy, or probabilistic flipping conditions.

A further limitation arises from the defender's perspective: Crossfire assumes the availability of secure storage for original honeypot weights and hash values, as well as access to unlabeled data to guide honeypot selection. While reasonable in controlled environments with trusted execution hardware, these requirements may be difficult to guarantee in distributed or resource-constrained deployments. Taken together, the two works explore complementary extremes of the threat landscape, with IBFA focusing on the maximal destructive potential of BFAs and Crossfire emphasizing defense under strong but idealized assumptions about storage integrity and monitoring infrastructure.

Numerical Representations. A central limitation across our works on BFAs is the reliance on specific numerical representations of GNN weights. IBFA and Crossfire (Paper 1 and Paper 2) explicitly operate on `int8`-quantized models, reflecting both the prevalence of quantization in efficiency-driven GNN applications and the fact that integer encodings bound the destructive potential of single bit flips compared to floating-point formats. While this framing is realistic for edge and embedded deployments, it excludes scenarios where GNNs are deployed in full-precision (`float32`) form, where exponent bit flips can cause disproportionately larger perturbations and have been shown to induce qualitatively different vulnerabilities. Our work on robustness bounds (Paper 3) extends this analysis by formalizing how expressivity degradation depends not only on architectural choices and graph properties but also on bit width, feature encoding, and activation functions, yet still assumes idealized, deterministic bit manipulations under specific representations. Collectively, these assumptions limit the generality of our findings: robustness and vulnerability patterns may shift under mixed-precision arithmetic, stochastic rounding, or hardware-dependent numerical faults.

Architectural Scope. Another limitation arises from the architectural focus of our studies. All three works primarily consider message-passing neural networks, with IBFA (Paper 1) and Paper 3 centering on GIN as representative of 1-WL expressive architectures, while Crossfire (Paper 2) evaluates defenses across a set of widely used MP-GNN variants. This choice reflects the dominance of MP-GNNs in practice and their well-understood expressivity, but it narrows the scope: higher-order GNNs [MRF+19], attention-based models [VCC+18], architectures incorporating non-standard aggregation functions [ESA+25] or hierarchical abstractions [FYW20] or graph transformers [MCB+22, YJK+19a] may exhibit qualitatively different vulnerabilities and defense dynamics. Moreover, MP-GNNs rely heavily on neighborhood aggregation, making them a natural target for expressivity-degrading attacks such as IBFA, whereas architectures departing from this

paradigm [YJK+19b] may require fundamentally different attack strategies. Thus, while our results establish principled insights for the MP-GNN family, their generalizability to more recent or specialized architectures remains an open question.

4.3. Future Work

Beyond the concrete contributions of Paper 1–Paper 4, the presented thesis points toward several broader research directions. At a high level, we identify three overarching goals for future research on expressive, efficient and robust graph neural networks.

First, there is a need for a unified view of robustness across perturbation spaces. In this thesis, we focused primarily on parameter-space attacks in the form of BFAs, but practical systems may be exposed simultaneously to graph poisoning and evasion attacks. Developing conceptual connections between these threats would clarify how different robustness mechanisms interact and where, as first indicated by [ESA+25], defenses can be shared or transferred.

Second, the thesis indicates a trade-off between efficiency, expressivity, and robustness. While quantization and sparsification may improve efficiency, they can as well reshape the model's expressivity and its sensitivity to BFAs. A systematic theoretical framework is needed to characterize how numerical representations, sparsity, activation functions, and graph structure jointly determine both expressivity and vulnerability, and to use these insights to design architectures that are simultaneously efficient, expressive, and robust.

Third, our current results are restricted to static, non-relational graphs. Many applications, however, are dynamic or multirelational by nature. Therefore, extending the results presented in this thesis, such as SELTH, to relational [BGMO22], temporal GNNs [WR25] or hierarchical message passing [FYW20], is essential for widening their applicability and consequential impact.

In the following paragraphs, we hence outline concrete research ideas that align with these broader goals.

Linking Poisoning, Evasion, and BFAs. Future research should examine the interplay between graph poisoning, evasion, and BFAs. Building on the notion of transfer learning [YXL22], we propose to explore *model evasion injection*, where the output of a clean model on a perturbed graph is transferred to a perturbed model under malicious bit flips on a clean graph. To ensure that the attack does not introduce excessive collateral effects, constraints such as bounding the Hamming distance[1] between the binarized clean and perturbed models [RHF19] should be imposed. This line of research would provide a more in-depth understanding of whether poisoning and evasion represent duals of BFAs, thereby clarifying their conceptual relationship.

[1]The Hamming distance between two equal-length binary strings is the number of bit positions at which they differ. In the BFA/PBS setting, this is computed layer-wise between the clean and perturbed binary weight tensors and then summed across layers to obtain a model-wide budget; it both constrains the attack and reports how many bits were effectively flipped [RHF19].

Attacker Budget Constraints. Our current studies (Paper 1 and Paper 2) assume an idealized attacker capable of executing arbitrary bit flips. However, real-world adversaries operate under strict budgetary constraints [HMDD22]. Future defenses should leverage this limitation by, for example, periodically permuting weight-input associations. If such permutations are applied every t_d time units while the attacker is required $t_a > t_d$ to compute and execute a solution, targeted flips will miss their intended weights, drastically reducing attack efficacy. This perspective introduces time and resource constraints as fundamental defense parameters.

Loss-Surface-Aware Defenses. Crossfire (Paper 2) demonstrated the feasibility of restoring GNNs after BFAs without retraining or labeled data. A natural extension is to exploit the geometry of the loss surface. By approximating curvature through diagonal Hessian estimates [DKH20] or Fisher information surrogates [SA20, WMS$^+$24], potentially using substitutes for only once differentiable activations [TMH$^+$23], one could identify vulnerable weights and steer gradient-based attackers toward flat regions where bit flips have minimal effect. Incorporating higher-order information into honeypot selection and pre-deployment pruning would enhance the resilience of GNNs while maintaining efficiency.

Efficiency-Robustness-Expressivity Trade-offs. Building on our preliminary expressivity-robustness analysis (Paper 3) future work should formalize how pruning and quantization interact with GNN resilience to BFAs. A theoretical framework is needed to derive bounds on the number of bit flips required to degrade expressivity under varying numerical representations (for example, `int8`, `float8`, or even lower bit widths [Cor25]), activation functions, and graph structural properties. These insights will guide the design of novel architectures that balance efficiency and robustness. For instance, replacing one-hot encodings with linearly independent dense embeddings or modifying quantization distributions may preserve expressivity while mitigating vulnerability.

Dynamic and Relational Settings. An important next step is to extend our analysis of BFAs and the SELTH (Paper 4) into dynamic and multirelational graph domains. While our current work focuses primarily on static, non-relational settings, recent advances in relational and temporal expressivity [BGMO22, WR25, HSMS25] suggest that sparsity-expressivity trade-offs may manifest differently when message passing interacts with relation-specific operators or time-respecting structures. Future research should therefore investigate whether winning tickets preserving 1-Relational WL (1-RWL) expressivity can likewise be identified in relational GNNs such as R-GCNs [SKB$^+$18] and CompGCNs [VSNT19], and whether sparsity can be leveraged to preserve temporal expressivity in temporal GNNs operating under causal constraints. Furthermore, integrating SELTH with higher-order and hierarchical message passing [FYW20] in dynamic and relational graphs could yield a unifying theory of expressive and efficient subnetworks that extends across static, temporal, and multirelational domains, thereby broadening the practical

impact of LTH-inspired methods in real-world applications such as knowledge graph reasoning, dynamic recommender systems, and molecular interaction networks.

Robustness of Winning Lottery Tickets to BFAs Building on the SELTH results in Paper 4 and the expressivity-robustness analysis in Paper 3, a natural next step is to test whether expressive winning tickets – that is, sparse initializations that preserve 1-WL-level expressivity – are intrinsically more robust to bit-flip attacks than alternative tickets at the same sparsity. Concretely, this entails (i) generating multiple tickets per sparsity via different pruning/initialization schemes (expressivity-preserving vs. baseline/random), (ii) quantifying their ex ante expressivity margin (for example, WL color separability proxies, injectivity diagnostics of MLPs), and (iii) benchmarking their degradation under IBFA (Paper 1) and PBFA across standard quantized and floating-point regimes, controlled bit-flip budgets, and data regimes varying in homophily, feature encoding (one-hot vs. dense), and activation choice (SiLU vs. ReLU). A complementary theoretical thread should extend the bounds in Paper 3 to sparse parameterizations, relating robustness to the distribution of critical computational paths retained by a ticket.

4.4. Final Remarks

For this thesis, we conducted the first systematic exploration of the robustness of GNNs to BFAs. To this end, we developed the first BFA specifically designed to exploit GNN expressivity, as well as a corresponding defense mechanism. Moreover, we provided a theoretical analysis of GNN robustness under BFAs and, with SELTH, introduced a novel formulation of the LTH for GNNs. Each of these contributions constitutes an innovation in its own right, and, taken together, they exemplify the intersection articulated in our research goals (Section 1.1): *efficiency* (via quantization and sparsity), *expressivity* (as both an architectural property and an attack surface), and *robustness* (to bit-level faults) are studied not in isolation but through their overlaps. In particular, we explore how strengthening one axis can reshape another, thereby providing a coherent blueprint for building GNNs that are simultaneously more efficient and more secure and thus more suitable for reliable deployment in science and industry.

Bibliography

[AGA+23] Tal Amir, Steven Gortler, Ilai Avni, Ravina Ravina, and Nadav Dym. Neural injective functions for multisets, measures and graphs via a finite witness theorem. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 42516–42551, 2023.

[AGKG22] Nurudin Alvarez-Gonzalez, Andreas Kaltenbrunner, and Vicenç Gómez. Beyond 1-WL with local ego-network encodings. In *Learning on Graphs Conference (LoG)*, 2022.

[ARM09] Architecure ARM. Security technology building a secure system using trustzone technology (white paper). *ARM Limited*, 2009.

[BÏ14] Elsbeth Bösl. The contergan scandal. media, medicine, and thalidomide in 1960s west germany. *Disability Histories*, pages 136–162, 2014.

[BBJP18] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. CSI neural network: Using side-channels to recover your artificial neural network information. *Cryptology ePrint Archive*, abs/2204.07697, 2018.

[BBZ21] Mehdi Bahri, Gaétan Bahl, and Stefanos Zafeiriou. Binary graph neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9492–9501, 2021.

[BF23] Blaž Bertalanič and Carolina Fortuna. Graph isomorphism networks for wireless link layer anomaly classification. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2023.

[BG20] Rickard Brüel Gabrielsson. Universal function approximation on graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 19762–19772, 2020.

[BGMO22] Pablo Barcelo, Mikhail Galkin, Christopher Morris, and Miguel Romero Orth. Weisfeiler and leman go relational. In *Learning on Graphs Conference (LoG)*, pages 46:1–46:26, 2022.

[BHJ+18] Jakub Breier, Xiaolu Hou, Dirmanto Jap, Lei Ma, Shivam Bhasin, and Yang Liu. Practical fault attack on deep neural networks. In *2018 ACM SIGSAC Conference on Computer and Communications Security*, page 2204–2206, 2018.

[BLC13] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013.

[CCZW21] Xuxi Chen, Tianlong Chen, Zhenyu Zhang, and Zhangyang Wang. You are caught stealing my winning lottery ticket! making a lottery ticket claim its ownership. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1780–1791, 2021.

[CM20] Mark Cheung and José MF Moura. Graph neural networks for covid-19 drug discovery. In *IEEE International Conference on Big Data*, pages 5646–5648, 2020.

[CMDLP23] Kamran Chitsaz, Goncalo Mordido, Jean-Pierre David, and François Leduc-Primeau. Training DNNs resilient to adversarial and random bit-flips by learning quantization ranges. *Transactions on Machine Learning Research*, 2023.

[Cor25] NVIDIA Corporation. *NVIDIA Blackwell Architecture Technical Overview*, 2025. Accessed: 2025-03-21.

[CSC+21] Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. A unified lottery ticket hypothesis for graph neural networks. In *International Conference on Machine Learning (ICML)*, pages 1695–1706, 2021.

[CVCB19] Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. In *Advances in neural information processing systems (NeurIPS)*, 2019.

[CW19] Chen Cai and Yusu Wang. A simple yet effective baseline for non-attributed graph classification. In *International Conference on Learning Representations (ICLR) Workshop on Representation learning*, 2019.

[dCNV22] Arthur da Cunha, Emanuele Natale, and Laurent Viennot. Proving the strong lottery ticket hypothesis for convolutional neural networks. In *International Conference on Learning Representations (ICLR)*, 2022.

[DKH20] Felix Dangel, Frederik Kunstner, and Philipp Hennig. BackPACK: Packing more into backprop. In *International Conference on Learning Representations (ICLR)*, 2020.

[DPSW+21] Austin Derrow-Pinion, Jennifer She, David Wong, Oliver Lange, Todd Hester, Luis Perez, Marc Nunkesser, Seongjae Lee, Xueying Guo, Brett Wiltshire, et al. Eta prediction with graph neural networks in google maps. In *ACM International Conference on Information & Knowledge Management*, pages 3767–3776, 2021.

[DTW+23] Guimin Dong, Mingyue Tang, Zhiyuan Wang, Jiechao Gao, Sikun Guo, Lihua Cai, Robert Gutierrez, Bradford Campbel, Laura E Barnes, and Mehdi Boukhechba. Graph neural networks in IoT: a survey. *ACM Transactions on Sensor Networks*, 19(2):1–50, 2023.

[DZZ+22] Enyan Dai, Tianxiang Zhao, Huaisheng Zhu, Junjie Xu, Zhimeng Guo, Hui Liu, Jiliang Tang, and Suhang Wang. A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability. *CoRR*, abs/2204.08570, 2022.

[ESA+25] Sofiane Ennadir, Oleg Smirnov, Yassine Abbahaddou, Lele Cao, and Johannes Lutzeyer. Enhancing graph classification robustness with singular pooling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[FC18] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[FDRC19] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Stabilizing the lottery ticket hypothesis. *CoRR*, abs/1903.01611, 2019.

[FL19] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch geometric. In *International Conference on Learning Representations (ICLR) Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[FMVG24] Billy Joe Franks, Christopher Morris, Ameya Velingker, and Floris Geerts. Weisfeiler-leman at the margin: When more expressivity matters. In *International Conference on Machine Learning (ICML)*, pages 13885–13926, 2024.

[FWL+20] Boyuan Feng, Yuke Wang, Xu Li, Shu Yang, Xueqiao Peng, and Yufei Ding. SGQuant: Squeezing the last bit on graph neural networks with specialized quantization. In *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, volume 39, pages 6347–6364. Springer, 2020.

[FYW20] Matthias Fey, Jan-Gin Yuen, and Frank Weichert. Hierarchical inter-message passing for learning on molecular graphs. In *International Conference on Machine Learning (ICML) Graph Representation Learning and Beyond (GRL+) Workshop*, 2020.

[GHYS22] Yun Gao, Hirokazu Hasegawa, Yukiko Yamaguchi, and Hajime Shimada. Malware detection by control-flow graph level representation learning

with graph isomorphism network. *IEEE Access*, 10:111830–111841, 2022.

[GLX+22] Jianliang Gao, Tengfei Lyu, Fan Xiong, Jianxin Wang, Weimao Ke, and Zhao Li. Predicting the survival of cancer patients with multimodal graph neural network. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 19(2):699–709, 2022.

[GMP21] Floris Geerts, Filip Mazowiecki, and Guillermo Perez. Let's agree to degree: Comparing graph convolutional networks in the message-passing framework. In *International Conference on Learning Representations (ICLR)*, pages 3640–3649, 2021.

[Gro21] Martin Grohe. The logic of graph neural networks. In *ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–17, 2021.

[HBJ+20] Xiaolu Hou, Jakub Breier, Dirmanto Jap, Lei Ma, Shivam Bhasin, and Yang Liu. Security evaluation of deep neural network resistance against laser fault injection. In *IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits*, pages 1–6, 2020.

[HFK+19] Sanghyun Hong, Pietro Frigo, Yigitcan Kaya, Cristiano Giuffrida, and Tudor Dumitras. Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks. In *USENIX Security Symposium*, pages 497–514, 2019.

[HFZ+20] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 22118–22133, 2020.

[HMDD22] Kevin Hector, Pierre-Alain Moëllic, Mathieu Dumont, and Jean-Max Dutertre. A closer look at evaluating the bit-flip attack against deep neural networks. In *IEEE 28th International Symposium on On-Line Testing and Robust System Design*, pages 1–5, 2022.

[HRL+20] Zhezhi He, Adnan Siraj Rakin, Jingtao Li, Chaitali Chakrabarti, and Deliang Fan. Defending and harnessing the bit-flip based adversarial weight attack. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14083–14091, 2020.

[HSMS25] Franziska Heeg, Jonas Sauer, Petra Mutzel, and Ingo Scholtes. Weisfeiler and leman follow the arrow of time: Expressive power of message passing in temporal event graphs. *CoRR*, abs/2505.24438, 2025.

[HY21] Jing Huang and Jie Yang. UniGNN: a unified framework for graph and hypergraph neural networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2563–2569, 2021.

[HYMK23] Bo Hui, Da Yan, Xiaolong Ma, and Wei-Shinn Ku. Rethinking graph lottery tickets: Graph sparsity matters. In *International Conference on Learning Representations (ICLR)*, 2023.

[HZRS16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[JK21] Mojan Javaheripi and Farinaz Koushanfar. HASHTAG: Hash signatures for online detection of fault-injection attacks on deep neural networks. In *IEEE/ACM International Conference On Computer Aided Design*, pages 1–9, 2021.

[JLX+21] Wei Jin, Yaxing Li, Han Xu, Yiqi Wang, Shuiwang Ji, Charu Aggarwal, and Jiliang Tang. Adversarial attacks and defenses on graphs. *ACM SIGKDD Explorations Newsletter*, 22(2):19–34, 2021.

[JWG+22] Di Jin, Rui Wang, Meng Ge, Dongxiao He, Xiang Li, Wei Lin, and Weixiong Zhang. RAW-GNN: random walk aggregation based graph neural network. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 2108–2114, 2022.

[JWL+22] Xun Jiao, Ruixuan Wang, Fred Lin, Daniel Moore, and Sriram Sankar. Assessing and analyzing the resilience of graph neural networks against hardware faults. *CoRR*, abs/2212.03475, 2022.

[Ker22] Nicolas Keriven. Not too little, not too much: a theoretical analysis of graph (over) smoothing. In *Advances in neural information processing systems (NeurIPS)*, pages 2268–2281, 2022.

[Kie20] Sandra Kiefer. *Power and limits of the Weisfeiler-Leman algorithm.* PhD thesis, Dissertation, RWTH Aachen University, 2020.

[KL51] Solomon Kullback and Richard Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22:79–86, 1951.

[KLI+22] Yash Khare, Kumud Lakara, Maruthi S. Inukonda, Sparsh Mittal, Mahesh Chandra, and Arvind Kaushik. Design and analysis of novel bit-flip attacks and defense strategies for DNNs. In *IEEE Conference on Dependable and Secure Computing*, pages 1–8, 2022.

[LK62] Widukind Lenz and Klais Knapp. Thalidomide embryopathy. *Archives of Environmental Health: An International Journal*, 5(2):14–19, 1962.

[LLP+20] Zheng Liu, Xiaohan Li, Hao Peng, Lifang He, and S Yu Philip. Heterogeneous similarity graph neural network on electronic health records. In *IEEE International Conference on Big Data*, pages 1196–1205, 2020.

[Lou20] Andreas Loukas. What graph neural networks cannot learn: Depth vs width. In *International Conference on Learning Representations (ICLR)*, 2020.

[LQZL20] Yang Li, Buyue Qian, Xianli Zhang, and Hui Liu. Graph neural network-based diagnosis prediction. *Big Data*, 8(5):379–390, 2020.

[LRH+21] Jingtao Li, Adnan Siraj Rakin, Zhezhi He, Deliang Fan, and Chaitali Chakrabarti. RADAR: Run-time adversarial weight attack detection and accuracy recovery. In *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pages 790–795, 2021.

[LRX+20] Jingtao Li, Adnan Siraj Rakin, Yan Xiong, Liangliang Chang, Zhezhi He, Deliang Fan, and Chaitali Chakrabarti. Defending bit-flip attack through DNN weight reconstruction. In *ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.

[LSR+20] Moritz Lipp, Michael Schwarz, Lukas Raab, Lukas Lamster, Misiker Tadesse Aga, Clémentine Maurice, and Daniel Gruss. Nethammer: Inducing rowhammer faults through network requests. In *IEEE European Symposium on Security and Privacy Workshops*, pages 710–719, 2020.

[LU21] Haohui Lu and Shahadat Uddin. A weighted patient network-based framework for predicting chronic diseases using graph neural networks. *Scientific reports*, 11(1):22607, 2021.

[LWLX17] Yannan Liu, Lingxiao Wei, Bo Luo, and Qiang Xu. Fault injection attack on deep neural network. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 131–138, 2017.

[LYW+23] Qi Liu, Jieming Yin, Wujie Wen, Chengmo Yang, and Shi Sha. NeuroPots: Realtime proactive defense against bit-flip attacks in neural networks. In *USENIX Security Symposium*, pages 6347–6364, 2023.

[LZH+24] Bohan Liu, Zijie Zhang, Peixiong He, Zhensen Wang, Yang Xiao, Ruimeng Ye, Yang Zhou, Wei-Shinn Ku, and Bo Hui. A survey of lottery ticket hypothesis. *CoRR*, abs/2403.04861, 2024.

[MAA+16] Frank McKeen, Ilya Alexandrovich, Ittai Anati, Dror Caspi, Simon Johnson, Rebekah Leslie-Hurd, and Carlos Rozas. Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave. In *Hardware and Architectural Support for Security and Privacy*, pages 1–9, 2016.

[MCB+22] Erxue Min, Runfa Chen, Yatao Bian, Tingyang Xu, Kangfei Zhao, Wenbing Huang, Peilin Zhao, Junzhou Huang, Sophia Ananiadou, and Yu Rong. Transformer for graphs: An overview from architecture perspective. *CoRR*, abs/2202.08455, 2022.

[MDM20] Jiaqi Ma, Shuangrui Ding, and Qiaozhu Mei. Towards more practical adversarial attacks on graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4756–4766, 2020.

[MFK21] Christopher Morris, Matthias Fey, and Nils Kriege. The power of the weisfeiler-leman algorithm for machine learning with graphs. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4543–4550, 2021.

[MGP+21] Shengjie Min, Zhan Gao, Jing Peng, Liang Wang, Ke Qin, and Bo Fang. Stgsn–a spatial–temporal graph neural network framework for time-evolving social networks. *Knowledge-Based Systems*, 214:106746, 2021.

[MK19] Onur Mutlu and Jeremie S Kim. Rowhammer: A retrospective. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(8):1555–1571, 2019.

[MKB+20] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A collection of benchmark datasets for learning with graphs. In *International Conference on Machine Learning (ICML) Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020.

[MKG25] Samir Moustafa, Nils Kriege, and Wilfried N Gansterer. Efficient mixed precision quantization in graph neural networks. In *IEEE International Conference on Data Engineering (ICDE)*, pages 4038–4052, 2025.

[MLM+23] Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M. Kriege, Martin Grohe, Matthias Fey, and Karsten Borgwardt. Weisfeiler and leman go machine learning: The story so far. *Journal of Machine Learning Research*, 24(333):1–59, 2023.

[MRF+19] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Conference on Artificial Intelligence (AAAI)*, pages 4602–4609, 2019.

[MYSSS20] Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning (ICML)*, pages 6682–6691, 2020.

[OL22] Ozan Özdenizci and Robert Legenstein. Improving robustness against stealthy weight bit-flip attacks by output code matching. In *Conference on Computer Vision and Pattern Recognition*, pages 13388–13397, 2022.

[PKL+22] Michael Puthawala, Konik Kothari, Matti Lassas, Ivan Dokmanić, and Maarten De Hoop. Globally injective relu networks. *Journal of Machine Learning Research*, 23(1):4544–4598, 2022.

[PS19] Sandro Pinto and Nuno Santos. Demystifying arm trustzone: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 51(6):1–36, 2019.

[PVMLSAMF15] Jaime Pérez-Villanueva, Oscar Méndez-Lucio, Olivia Soria-Arteche, and José L. Medina-Franco. Activity cliffs and activity cliff generators based on chemotype-related activity landscapes. *Molecular Diversity*, 19(4):1021–1035, 2015.

[QZN+23] Cheng Qian, Ming Zhang, Yuanping Nie, Shuaibing Lu, and Huayang Cao. A survey of bit-flip attacks on deep neural network and corresponding defense methods. *Electronics*, 12(4):853, 2023.

[RHF19] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. Bit-flip attack: Crushing neural network with progressive bit search. In *International Conference on Computer Vision (ICCV)*, pages 1211–1220, 2019.

[RHL+22] Adnan Siraj Rakin, Zhezhi He, Jingtao Li, Fan Yao, Chaitali Chakrabarti, and Deliang Fan. T-BFA: Targeted bit-flip adversarial weight attack. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7928–7939, 2022.

[RJK+20] Ryan A Rossi, Di Jin, Sungchul Kim, Nesreen K Ahmed, Danai Koutra, and John Boaz Lee. On proximity and structural role-based embeddings in networks: Misconceptions, techniques, and applications. *ACM Transactions on Knowledge Discovery from Data*, 14(5):1–37, 2020.

[SA20] Sidak Pal Singh and Dan Alistarh. WoodFisher: Efficient second-order approximation for neural network compression. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 18098–18109, 2020.

[SJN21] Tailor Shyam, Fernandez-Marques Javier, and Lane Nicholas. Degree-quant: Quantization-aware training for graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2021.

[SKB+18] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607, 2018.

[SLG+24] Yingxia Shao, Hongzheng Li, Xizhi Gu, Hongbo Yin, Yawen Li, Xupeng Miao, Wentao Zhang, Bin Cui, and Lei Chen. Distributed graph neural network training: A survey. *ACM Computing Surveys*, 56(8):1–39, 2024.

[SWC+23] Yongduo Sui, Xiang Wang, Tianlong Chen, Meng Wang, Xiangnan He, and Tat-Seng Chua. Inductive lottery ticket learning for graph neural networks. *Journal of Computer Science and Technology*, pages 1223–1237, 2023.

[SWT+20] Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In *Web Conference 2020*, page 673–683, 2020.

[SYC+21] Zhenchao Sun, Hongzhi Yin, Hongxu Chen, Tong Chen, Lizhen Cui, and Fan Yang. Disease prediction via graph neural networks. *IEEE Journal of Biomedical and Health Informatics*, 25(3):818–826, 2021.

[TMH+23] Max Torop, Aria Masoomi, Davin Hill, Kivanc Kose, Stratis Ioannidis, and Jennifer Dy. SmoothHess: Relu network feature interactions via stein's lemma. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 50697–50729, 2023.

[TP23] Anton Tsitsulin and Bryan Perozzi. The graph lottery ticket hypothesis: Finding sparse, informative graph structure. *CoRR*, abs/2312.04762, 2023.

[VCC+18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[VMA+20] Valerio Venceslai, Alberto Marchisio, Ihsen Alouani, Maurizio Martina, and Muhammad Shafique. Neuroattack: Undermining spiking neural networks security through externally triggered bit-flips. In *International Joint Conference on Neural Networks (IJCAI)*, pages 1–8, 2020.

[VSNT19] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. *CoRR*, abs/1911.03082, 2019.

[WCP+22] Lingfei Wu, Peng Cui, Jian Pei, Liang Zhao, and Le Song. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer, 2022.

[WK16] Max Welling and Thomas N Kipf. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2016.

[WKH+24] Tom Wollschläger, Niklas Kemper, Leon Hetzel, Johanna Sommer, and Stephan Günnemann. Expressivity and generalization: Fragment-biases for molecular gnns. In *International Conference on Machine Learning (ICML)*, pages 53113–53139, 2024.

[WL68] Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *nti, Series*, 2(9):12–16, 1968.

[WLL+23] Zhiqiong Wang, Zican Lin, Shuo Li, Yibo Wang, Weiying Zhong, Xinlei Wang, and Junchang Xin. Dynamic multi-task graph isomorphism network for classification of alzheimer's disease. *Applied Sciences*, 13(14):8433, 2023.

[WLW+22] Kun Wang, Yuxuan Liang, Pengkun Wang, Xu Wang, Pengfei Gu, Junfeng Fang, and Yang Wang. Searching lottery tickets in graph neural networks: A dual perspective. In *International Conference on Learning Representations (ICLR)*, 2022.

[WMS+24] Diyuan Wu, Ionut-Vlad Modoranu, Mher Safaryan, Denis Kuznedelev, and Dan Alistarh. The iterative optimal brain surgeon: Faster sparse recovery by leveraging second-order information. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 139621–139649, 2024.

[WR25] Przemysław Andrzej Wałęga and Michael Rawson. Expressive power of temporal message passing. In *Conference on Artificial Intelligence (AAAI)*, pages 21000–21008, 2025.

[WRF+18] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[WSZ+22] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: A survey. *ACM Computing Surveys*, 55(5):1–37, 2022.

[WYW+23] Bang Wu, Xingliang Yuan, Shuo Wang, Qi Li, Minhui Xue, and Shirui Pan. Securing graph neural networks in mlaas: A comprehensive realization of query-based integrity verification. *CoRR*, abs/2312.07870, 2023.

[WYW+24] Bang Wu, Xingliang Yuan, Shuo Wang, Qi Li, Minhui Xue, and Shirui Pan. Securing graph neural networks in mlaas: A comprehensive realisation of query-based integrity verification. In *IEEE Symposium on Security and Privacy (SP)*, pages 110–110, 2024.

[WZW+23] Jialai Wang, Ziyuan Zhang, Meiqi Wang, Han Qiu, Tianwei Zhang, Qi Li, Zongpeng Li, Tao Wei, and Chao Zhang. Aegis: Mitigating targeted bit-flip attacks against deep neural networks. In *USENIX Security Symposium*, pages 2329–2346, 2023.

[XHLJ19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

[xO25a] ASAP Discovery x OpenADMET. Antiviral ADMET 2025. Data set, 2025. License: CC0-1.0.

[xO25b] ASAP Discovery x OpenADMET. Antiviral potency 2025. Data set, 2025. License: CC-BY-4.0.

[XXC+21] Jiacheng Xiong, Zhaoping Xiong, Kaixian Chen, Hualiang Jiang, and Mingyue Zheng. Graph neural networks for automated de novo drug design. *Drug Discovery Today*, 26(6):1382–1393, 2021.

[XZF+21] Jingjing Xu, Wangchunshu Zhou, Zhiyi Fu, Hao Zhou, and Lei Li. A survey on green deep learning. *CoRR*, abs/2111.05193, 2021.

[YFT20] Mengjia Yan, Christopher Fletcher, and Josep Torrellas. Cache telepathy: Leveraging shared resource attacks to learn DNN architectures. In *USENIX Security Symposium*, pages 2003–2020, 2020.

[YHS+22] Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 1287–1292, 2022.

[YIGA+24] Jiale Yan, Hiroaki Ito, Ángel López Garcıá-Arias, Yasuyuki Okoshi, Hikari Otsuka, Kazushi Kawamura, Thiem Van Chu, and Masato Motomura. Multicoated and folded graph neural networks with strong lottery tickets. In *Learning on Graphs Conference (LoG)*, pages 11–1, 2024.

[YJK+19a] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. In *Advances in neural information processing systems (NeurIPS)*, 2019.

[YJK+19b] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 11983–11993, 2019.

[YJLH22] Sihong Yang, Dezhi Jin, Jun Liu, and Ye He. Identification of young high-functioning autism individuals based on functional connectome using graph isomorphism network: A pilot study. *Brain Sciences*, 12(7):883, 2022.

[YPL+18] Dong Yin, Ashwin Pananjady, Max Lam, Dimitris Papailiopoulos, Kannan Ramchandran, and Peter Bartlett. Gradient diversity: a key ingredient for scalable distributed learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1998–2007, 2018.

[YRF20] Fan Yao, Adnan Siraj Rakin, and Deliang Fan. Deephammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips. In *USENIX Security Symposium*, pages 1463–1480, 2020.

[YXJ+24] Wang Yuxin, Hu Xiannian, Xie Jiaqing, Yin Zhangyue, Zhou Yunhua, Qiu Xipeng, and Huang Xuanjing. Graph structure learning via lottery hypothesis at scale. In *Asian Conference on Machine Learning*, pages 1401–1416, 2024.

[YXL22] Fuchao Yu, Xianchao Xiu, and Yunhui Li. A survey on deep transfer learning and beyond. *Mathematics*, 10(19):3619, 2022.

[YZY+22] Jiangchao Yao, Shengyu Zhang, Yang Yao, Feng Wang, Jianxin Ma, Jianwei Zhang, Yunfei Chu, Luo Ji, Kunyang Jia, Tao Shen, et al. Edge-cloud polarization and collaboration: A comprehensive survey for ai. *IEEE Transactions on Knowledge and Data Engineering*, 35(7):6866–6886, 2022.

[ZAG18] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, page 2847–2856, 2018.

[ZCS+22] Sixiao Zhang, Hongxu Chen, Xiangguo Sun, Yicong Li, and Guandong Xu. Unsupervised graph poisoning attack via contrastive loss back-propagation. In *ACM Web Conference 2022*, pages 1322–1330, 2022.

[ZCSL19] Jiaqi Zhang, Xiangru Chen, Mingcong Song, and Tao Li. Eager pruning: Algorithm and architecture support for fast training of deep neural networks. In *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, pages 292–303, 2019.

[ZG19] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations (ICLR)*, 2019.

[ZJL+22] Jiong Zhu, Junchen Jin, Donald Loveland, Michael T Schaub, and Danai Koutra. How does heterophily impact the robustness of graph neural networks? theoretical connections and practical implications. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2637–2647, 2022.

[ZJZ+21] Zeru Zhang, Jiayin Jin, Zijie Zhang, Yang Zhou, Xin Zhao, Jiaxiang Ren, Ji Liu, Lingfei Wu, Ruoming Jin, and Dejing Dou. Validating the lottery ticket hypothesis with inertial manifold theory. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 30196–30210, 2021.

[ZKR+17] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep sets. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3391–3401, 2017.

[ZLC24] Yilun Zheng, Sitao Luan, and Lihui Chen. What is missing in homophily? disentangling graph homophily for graph neural networks. In *Advances in neural information processing systems (NeurIPS)*, pages 68406–68452, 2024.

[ZLM+23] Zeyu Zhu, Fanrong Li, Zitao Mo, Qinghao Hu, Gang Li, Zejian Liu, Xiaoyao Liang, and Jian Cheng. A^2Q: Aggregation-aware quantization for graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2023.

[Zop22] Markus Zopf. 1-WL expressiveness is (almost) all you need. In *International Joint Conference on Neural Networks, (IJCNN)*, pages 1–8, 2022.

[ZWC+22] Ziyuan Zhang, Meiqi Wang, Wencheng Chen, Han Qiu, and Meikang Qiu. Mitigating targeted bit-flip attacks via data augmentation: An empirical study. In *Knowledge Science, Engineering and Management: 15th International Conference*, pages 606–618, 2022.

[ZWH+24] Guibin Zhang, Kun Wang, Wei Huang, Yanwei Yue, Yang Wang, Roger Zimmermann, Aojun Zhou, Dawei Cheng, Jin Zeng, and Yuxuan Liang. Graph lottery ticket automated. In *International Conference on Learning Representations (ICLR)*, 2024.

[ZWL+21] Shuai Zhang, Meng Wang, Sijia Liu, Pin-Yu Chen, and Jinjun Xiong. Why lottery ticket wins? a theoretical perspective of sample complexity on sparse neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2707–2720, 2021.

A. Appended Papers

The subsequent sections of this dissertation consist of the published articles that form the core contributions of this work, accompanied by supplementary material. For each article, we include the bibliographic details, such as title, abstract, publication venue and associated Computing Research and Education (CORE) 2023 ranking[1], list of authors, a statement on the distribution of contributions, and the associated Digital Object Identifier (DOI) or International Standard Serial Number (ISSN).

[1] `https://portal.core.edu.au/conf-ranks/`

A.1 Attacking Graph Neural Networks with Bit Flips

Title	Attacking Graph Neural Networks with Bit Flips: Weisfeiler and Leman Go Indifferent
Authors	Lorenz Kummer, Samir Moustafa, Sebastian Schrittwieser, Wilfried Gansterer, Nils Kriege
Venue	Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (CORE ranking: A^*)
Identifier	`DOI:10.1145/3637528.3671890`
Division of work	Lorenz Kummer: Conceived the initial idea; formulated the attack method and parts of its theoretical foundation; implemented most components and designed experiments; conducted the evaluation; wrote most of the article and created the visualizations; produced the conference poster and oral-presentation slides; reviewed the related work. Samir Moustafa: Implemented the underlying quantization package; participated in regular quantization discussions and occasional pair-programming sessions; wrote the quantization-related sections; reviewed and refined the article, conference poster, and slides. Sebastian Schrittwieser: Provided security domain expertise; refined the threat model and assumptions about attacker capabilities; reviewed and refined the article, conference poster, and slides. Wilfried Gansterer: Participated in discussions; reviewed and refined the article, conference poster, and slides; provided mentoring and supervision. Nils Kriege: Contributed some of the theoretical derivations presented in the article; participated in discussions; reviewed and refined the article – particularly its theoretical underpinnings – as well as the conference poster and slides; provided mentoring and supervision.

Attacking Graph Neural Networks with Bit Flips: Weisfeiler and Leman Go Indifferent

Lorenz Kummer
Doctoral School Computer Science,
Faculty of Computer Science,
University of Vienna
Vienna, Austria
lorenz.kummer@univie.ac.at

Samir Moustafa
Doctoral School Computer Science,
Faculty of Computer Science,
University of Vienna
Vienna, Austria
samir.moustafa@univie.ac.at

Sebastian Schrittwieser
Christian Doppler Laboratory for
Assurance and Transparency in
Software Protection,
Faculty of Computer Science,
University of Vienna
Vienna, Austria
sebastian.schrittwieser@univie.ac.at

Wilfried Gansterer
Faculty of Computer Science,
University of Vienna
Vienna, Austria
wilfried.gansterer@univie.ac.at

Nils Kriege
Research Network Data Science,
Faculty of Computer Science,
University of Vienna
Vienna, Austria
nils.kriege@univie.ac.at

ABSTRACT

Prior attacks on graph neural networks have focused on graph poisoning and evasion, neglecting the network's weights and biases. For convolutional neural networks, however, the risk arising from bit flip attacks is well recognized. We show that the direct application of a traditional bit flip attack to graph neural networks is of limited effectivity. Hence, we discuss the Injectivity Bit Flip Attack, the first bit flip attack designed specifically for graph neural networks. Our attack targets the learnable neighborhood aggregation functions in quantized message passing neural networks, degrading their ability to distinguish graph structures and impairing the expressivity of the Weisfeiler-Leman test. We find that exploiting mathematical properties specific to certain graph neural networks significantly increases their vulnerability to bit flip attacks. The Injectivity Bit Flip Attack can degrade the maximal expressive Graph Isomorphism Networks trained on graph property prediction datasets to random output by flipping only a small fraction of the network's bits, demonstrating its higher destructive power compared to traditional bit flip attacks transferred from convolutional neural networks. Our attack is transparent, motivated by theoretical insights and confirmed by extensive empirical results.

CCS CONCEPTS

• **Security and privacy** → **Security in hardware**; **Malware and its mitigation**; • **Computing methodologies** → *Neural networks*; • **Mathematics of computing** → Graph algorithms.

KEYWORDS

Bit Flip Attacks, Graph Neural Networks

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08.
https://doi.org/10.1145/3637528.3671890

ACM Reference Format:
Lorenz Kummer, Samir Moustafa, Sebastian Schrittwieser, Wilfried Gansterer, and Nils Kriege. 2024. Attacking Graph Neural Networks with Bit Flips: Weisfeiler and Leman Go Indifferent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671890

1 INTRODUCTION

Graph neural networks (GNNs) are a powerful machine learning technique for handling structured data represented as graphs with nodes and edges. These methods are highly versatile, extending the applicability of deep learning to new domains such as financial and social network analysis, medical data analysis, and chem- and bioinformatics [8, 16, 39, 56, 64, 65]. With the increasing adoption of GNNs, there is a pressing need to investigate their potential security vulnerabilities. Traditional adversarial attacks on GNNs have focused on manipulating input graph data [63] through poisoning attacks, which result in the learning of a faulty model [40, 63], or on evasion attacks, which use adversarial examples to degrade inference. These attacks can be targeted [78] or untargeted [40, 79] and involve modifications of node features, edges, or the injection of new nodes [55, 63]. Targeted attacks degrade the model's performance on a subset of instances, while untargeted attacks degrade the model's overall performance [75]. A classification of existing graph poisoning and evasion attacks and defense mechanisms as well as a repository with representative algorithms can be found in the comprehensive reviews by Jin et al. [28] and Dai et al. [9].

Previous research has shown that convolutional neural networks (CNNs) prominent in the computer vision domain are highly susceptible to Bit Flip Attacks (BFAs) [21, 47]. These works typically focus on CNNs to which quantization is applied, e.g., [47, 48], a common technique to improve efficiency. Quantized CNNs are naturally more perturbation resistant compared to their unquantized (i.e., floating-point) counterparts [21, 49], and thus their degradation poses a more challenging problem. For example, quantization

is mentioned by Hong et al. [21] as a viable protection mechanism against random BFAs. The perturbation resistance of quantized CNNs largely arises from their numerical representation. Specifically, using integer representations makes it difficult for a single bit flip to cause a significant increase in absolute value. For instance, in an 8-bit integer quantized format, the maximum increase a parameter can undergo due to a single bit flip is limited to 128. A flip in the exponent of an IEEE754 32-bit floating-point representation, however, can increase the parameter by up to $3.4 \cdot 10^{38}$, causing extreme neuron activation overriding the rest of the activations [21, 37]. Efficient implementation is likewise crucial for practical applications of GNNs, making it necessary to investigate the interaction between robustness and efficiency. The practical relevance of quantizing GNNs has been recognized for applications such as inference on IoT and edge devices [72], reducing energy consumption for green deep learning [67], and in distributed GNN applications [53]. Recent efforts have been made to further advance quantization techniques for GNNs [3, 13, 76] as well as technical realizations for their deployment [58, 76] and further potential applications exist, e.g., [6, 10, 11].

Prior research on the security of GNNs has mostly overlooked the potential threat of BFAs [28, 40, 63, 66] which directly manipulate a target model at inference time, and previous work on BFAs has not yet explored attacks on quantized GNNs [47]. Despite the known robustness of quantized CNNs to BFAs [21, 49], which potentially transfers to GNNs as it is mostly rooted in the numerical representation itself, the only investigations of GNN resilience to BFAs study bit flips in floating-point representation [27, 62]. Furthermore, while general BFAs may be transferable to GNNs, they do not consider the unique mathematical properties of GNNs, although it has been observed that adapting BFAs to the specific properties of a target network can increase harm [59] and that BFAs can be far from optimal on non-convolutional models [20]. We address this research gap by exploring the effects of malicious perturbations of the trainable parameters of quantized GNNs and their impact on prediction quality. Specifically, we target the expressivity of GNNs, i.e., their ability to distinguish non-isomorphic graphs or node neighborhoods. The most expressive GNNs based on standard message passing, including the widely-used Graph Isomorphism Networks (GINs) [68], have the same discriminative power as the 1-Weisfeiler-Leman test (1-WL) for suitably parameterized neighborhood aggregation functions [43, 44, 68]. Our attack targets the quantized parameters of these neighborhood aggregation functions to degrade the network's ability to differentiate between non-isomorphic structures.

1.1 Related work

The security issue of BFAs has been recognized for quantized CNNs, which are used in critical applications like medical image segmentation [2, 74] and diagnoses [18, 50]. In contrast, the robustness of GNNs used in safety-critical domains, like medical diagnoses [16, 33, 39], electronic health record modeling [38, 56], or drug development [8, 34, 65], against BFAs has not been sufficiently studied. Qian et al. [47] and Khare et al. [29] distinguish between targeted and untargeted BFAs similar to the distinction made between targeted and untargeted poisoning and evasion techniques. The high volume of related work on BFAs for quantized CNNs [29, 47] based on the seminal work by Rakin et al. [48] and associated BFA defense mechanisms, e.g., [29, 32, 36] published recently, underscores the need for research in the direction of both BFA and defense mechanisms for quantized GNNs. As a representative for these traditional BFA methods we focus on Progressive Bit Flip Attack [48] as most other BFA variants are based on it. We subsequently refer to this specific BFA as **PBFA** and use the term BFA for the broader class of bit flip attacks only. Existing research on the resilience of GNNs to bit flips focuses exclusively on floating-point representation: Jiao et al. [27] study random bit faults and Wu et al. [62] consider a variant of PBFA modified to flip bits in the exponent of a weight's floating-point representation.

1.2 Contribution

In a motivating case study, we explore quantized GNNs' vulnerability to PBFA, finding that PBFA does not notably surpass random bit flips in tasks demanding graph structural discrimination. To demonstrate that degrading such GNNs is nonetheless possible in the identified scenario, we introduce the Injectivity Bit Flip Attack (**IBFA**), a novel attack targeting the discriminative power of neighborhood aggregation functions used by GNNs. Specifically, we investigate the GIN architecture with 1-WL expressivity, where this function is injective for suitable parameters [68], and which is integrated in popular frameworks [14] and widely used in practice, e.g., [6, 17, 61, 70]. IBFA, unlike existing BFAs for CNNs, has a unique bit-search algorithm optimization goal and input data selection strategy. It differs from graph poisoning and evasion attacks as it leaves input data unmodified. We establish a theoretical basis for IBFA, and support it by empirical evidence of its efficacy on real-world datasets. IBFA outperforms baselines in degrading prediction quality and requires fewer bit flips to reduce GIN's output to randomness, making it indifferent to graph structures.

2 PRELIMINARIES

IBFA targets quantized neighborhood aggregation-based GNNs, exploiting their expressivity linked to the 1-WL graph isomorphism test. We begin by summarizing these concepts and introduce our notation and terminology along the way.

2.1 Graph theory

A *graph* G is a pair (V, E) of a finite set of *nodes* V and *edges* $E \subseteq \{\{u, v\} \subseteq V\}$. The set of nodes and edges of G is denoted by $V(G)$ and $E(G)$, respectively. The *neighborhood* of v in $V(G)$ is $N(v) = \{u \in V(G) \mid \{v, u\} \in E(G)\}$. If there exists a bijection $\varphi: V(G) \to V(H)$ with $\{u, v\}$ in $E(G)$ if and only if $\{\varphi(u), \varphi(v)\}$ in $E(H)$ for all u, v in $V(G)$, we call the two graphs G and H *isomorphic* and write $G \simeq H$. For two graphs with roots $r \in V(G)$ and $r' \in V(H)$, the bijection must additionally satisfy $\varphi(r) = r'$. The equivalence classes induced by $\simeq$ are referred to as *isomorphism types*.

A function $V(G) \to \Sigma$ is called a *node coloring*. Then, a *node colored* or *labeled graph* (G, l) is a graph G endowed with a node coloring l. We call $l(v)$ a *label* or *color* of $v \in V(G)$. We denote a multiset by $\{\!\{\ldots\}\!\}$.

2.2 The Weisfeiler-Leman algorithm

Let (G, l) denote a labelled graph. In every iteration $t > 0$, a node coloring $c_l^{(t)}\colon V(G) \longrightarrow \Sigma$ is computed, which depends on the coloring $c_l^{(t-1)}$ of the previous iteration. At the beginning, the coloring is initialized as $c_l^{(0)} = l$. In subsequent iterations $t > 0$, the coloring is updated according to

$$c_l^{(t)}(v) = \textsc{Hash}\left(c_l^{(t-1)}(v), \{\!\{c_l^{(t-1)}(u) | u \in N(v)\}\!\}\right), \tag{1}$$

where Hash is an injective mapping of the above pair to a unique value in Σ, that has not been used in previous iterations. The Hash function can, for example, be realized by assigning new consecutive integer values to pairs when they occur for the first time [54]. Let $C_l^{(t)}(G) = \{\!\{c_l^{(t)}(v) \mid v \in V(G)\}\!\}$ be the multiset of colors a graph exhibits in iteration t. The iterative coloring terminates if $|C_l^{(t-1)}(G)| = |C_l^{(t)}(G)|$, i.e., the number of colors does not change between two iterations. For testing whether two graphs G and H are isomorphic, the above algorithm is run in parallel on both G and H. If $C_l^{(t)}(G) \neq C_l^{(t)}(H)$ for any $t \geq 0$, then G and H are not isomorphic. The label assigned to a node v in the tth iteration of the 1-WL test can be understood as a tree representation of the t-hop neighborhood of v, in the sense that each $c_l^{(t)}(v)$ corresponds to an isomorphism type of such trees of height t, see Definition 3.1 and [12, 26, 52] for details.

2.3 Graph neural networks

Contemporary GNNs employ a neighborhood aggregation or message passing approach, in which the representation of a node is iteratively updated through the aggregation of representations of its neighboring nodes. Upon completion of k iterations of aggregation, the representation of a node encapsulates the structural information within its k-hop neighborhood [68]. The kth layer of a GNN computes the node features $\mathbf{h}_v^{(k)}$ formally defined by

$$\begin{aligned} \mathbf{a}_v^{(k)} &= \textsc{Aggregate}^{(k)}\left(\{\!\{\mathbf{h}_u^{(k-1)} \mid u \in N(v)\}\!\}\right), \\ \mathbf{h}_v^{(k)} &= \textsc{Combine}^{(k)}\left(\mathbf{h}_v^{(k-1)}, \mathbf{a}_v^{(k)}\right). \end{aligned} \tag{2}$$

Initially, $\mathbf{h}_v^{(0)}$ are the graph's node features. For graph level readout, individual node embeddings are aggregated into a single representation $\mathbf{h}_G$ of the entire graph

$$\mathbf{h}_G = \textsc{ReadOut}\left(\{\!\{\mathbf{h}_v^{(k)} \mid v \in G\}\!\}\right). \tag{3}$$

The choice of $\textsc{Aggregate}^{(k)}$, $\textsc{Combine}^{(k)}$ and ReadOut in GNNs is critical, and several variants have been proposed [68].

2.4 Graph isomorphism network

GIN has the same discriminative power as the 1-WL test in distinguishing non-isomorphic graphs [68]. A large body of work is devoted to GNNs exceeding this expressivity [43]. However, neighborhood aggregation is widely used in practice and 1-WL is sufficient to distinguish most graphs in common benchmark datasets [41, 77]. As established by Xu et al. [68], a neighborhood aggregation GNN with a sufficient number of layers can reach the same discriminative power as the 1-WL test if both the Aggregate and Combine functions in each layer's update rule as well as its graph level ReadOut are injective. GIN achieves this with the update rule

$$\mathbf{h}_v^{(k)} = \text{MLP}^{(k)}\left((1 + \epsilon^{(k)}) \cdot \mathbf{h}_v^{(k-1)} + \sum_{u \in N(v)} \mathbf{h}_u^{(k-1)}\right) \tag{4}$$

integrating a multi layer perceptron (MLP) into $\textsc{Combine}^{(k)}$, realizing a universal function approximator on multisets [22, 23, 73]. If input features are one-hot encodings, an MLP is not needed before summation in the first layer, since summation is injective in this case. Graph level readout in GIN is realized via first summing each layer's embeddings followed by concatenation (denoted by $\|$) of the summed vectors extracted from GINs' layers. The resulting graph level embedding $\mathbf{h}_G$ can then be used as input for, e.g., an MLP for subsequent graph level classification tasks.

$$\mathbf{h}_G = \Big\|_{k=0}^{n}\left(\sum_{v \in V(G)} \mathbf{h}_v^{(k)}\right) \tag{5}$$

2.5 Quantization

Quantization involves either a reduction in the precision of the weights, biases, and activations within a neural network or the use of a more efficient representation, resulting in a decrease in model size and memory utilization [31]. In accordance with the typical set-up chosen in the related work on BFA, see Section 1.1, we apply scale quantization to map `FLOAT32` tensors to the `INT8` range. Such a quantization function Q and its associated dequantization function Q^{-1} are

$$\begin{aligned} Q(\mathbf{W}^{(l)}) &= \mathbf{W}_q^{(l)} = \text{clip}(\lfloor \mathbf{W}^{(l)}/s \rceil,\ a, b), \\ Q^{-1}(\mathbf{W}_q^{(l)}) &= \widehat{\mathbf{W}}^{(l)} = \mathbf{W}_q^{(l)} \times s. \end{aligned} \tag{6}$$

Here, the variable s denotes the scaling parameter, $\text{clip}(x, a, b) = \min(\max(x, a), b)$ with a and b the minimum and maximum thresholds (also known as the quantization range), $\lfloor \dots \rceil$ denotes nearest integer rounding, $\mathbf{W}^{(l)}$ is the weight of a layer l to be quantized, $\mathbf{W}_q^{(l)}$ its quantized counterpart and $\widehat{\dots}$ indicates a perturbation, i.e., rounding errors in the case of Equation (6). Similar to other works on BFA that require quantized target networks, e.g., [48], we address the issue of non-differentiable rounding and clipping functions in Q by employing Straight Through Estimation (STE) [5].

2.6 Progressive bit flip attack

PBFA [48] uses a quantized trained CNN Φ and employs progressive bit search (**PBS**) to identify which bit flips will damage the CNN most. PBS begins with a single forward and backward pass, conducting error backpropagation without weight updates on a randomly selected batch $\mathbf{X}$ of training data with a target vector $\mathbf{t}$. It then selects the weights linked to the top-k largest binary encoded gradients as potential candidates for flipping the associated bits. These candidate bits are iteratively tested (flipped) across all L layers to find the bit that maximizes the difference between the loss $\mathcal{L}$ of the perturbed and the loss of the unperturbed CNN, whereby the same loss function is used that was minimized during training,

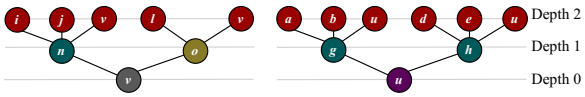


Figure 1: Example of two non-isomorphic unfolding trees $T^{(2)}(u) \not\simeq T^{(2)}(v)$ of height 2 associated with the nodes u and v. A function solving a WL-discrimination task for $k = 2$ must be able to discriminate u and v based on the structure of their unfolding trees.

e.g., (binary) cross entropy (CE) for (binary) classification.

$$\max_{\{\widehat{\mathbf{W}}_q^{(l)}\}} \quad \mathcal{L}\Big(\Phi(\mathbf{X};\ \{\widehat{\mathbf{W}}_q^{(l)}\}_{l=1}^{L}),\ \mathbf{t}\Big) - \mathcal{L}\Big(\Phi(\mathbf{X};\ \{\mathbf{W}_q^{(l)}\}_{l=1}^{L}),\ \mathbf{t}\Big) \tag{7}$$

The source of the perturbation $\widehat{\dots}$ in Equation (7) are adversarial bit flips. If a single bit flip does not improve the optimization goal, PBS is executed again, considering combinations of two or more bit flips. This process continues until the desired level of network degradation is achieved or a predefined number of iterations is reached. Further details on PBFA/PBS can be found in the appendix.

3 INJECTIVITY BIT FLIP ATTACK

In principle, PBFA and potentially most other BFA variants based on it can be directly ported to GNNs. However, Hector et al. [20] demonstrate that the high susceptibility of CNNs to BFA is closely tied to weight sharing in convolutional filters. Further, Hector et al. [20] show that MLPs are inherently more robust to PBFA than CNNs due to their different gradient distributions. The absence of convolutional filters in GNNs, as well as MLPs being an integral component of certain GNNs such as GIN, motivates the development of a specialized attack for GNNs.

In our preliminary case study (see appendix) we examine PBFA's effectiveness on various GNN architectures. The case study's results indicate that quantized GINs trained on tasks requiring discrimination of graphs based on their structural properties and thus high structural expressivity as found in, e.g., drug development, display a remarkable resilience to PBFA, which in some instances hardly outperformed random bit flips even after more than $2 \cdot 10^3$ bit flips. Considering contemporary literature [60, 71] postulates an upper limit of 24 to 50 precise bit flips to be achievable by an attacker within a span of several hours, the execution of such a substantial number of bit flips appears impractical.

Based on these observations, IBFA focuses on degrading GIN trained on tasks requiring high structural expressivity. The discriminative power of GIN is derived from its MLPs' ability to represent injective functions on sets (Section 2.4). Consequently, we design our attack assuming that in tasks where learning such a discriminative function is crucial, attacking injectivity will lead to a higher degradation than performing PBFA.

3.1 Expressivity via injective set functions

A GNN based on message passing computing a function $F^{(k)}$ as output of the kth layer is maximal expressive if $F^{(k)}(u) = F^{(k)}(v) \iff c_l^{(k)}(u) = c_l^{(k)}(v)$. This is achieved when each layers' Combine and Aggregate functions are both injective, such that their combination is injective as well.

We develop the theory behind a bit flip attack exploiting this property. We formally define the concept of an unfolding tree also known as *computational tree* in the context of GNNs [12, 26], see Figure 1 for an example.

Definition 3.1 (Unfolding Tree [12]). The *unfolding tree* $T^{(k)}(v)$ of height k of a vertex v is defined recursively as

$$T^{(k)}(v) = \begin{cases} \textsc{Tree}(l(v)) & \text{if } k = 0, \\ \textsc{Tree}(l(v), T^{(k-1)}(N(v))) & \text{if } k > 0, \end{cases}$$

where $\textsc{Tree}(l(v))$ is a tree containing a single node with label $l(v)$ and $\textsc{Tree}(l(v), T^{(k-1)}(N(v)))$ is a tree with a root node labeled $l(v)$ having the roots of the trees $T^{(k-1)}(N(v)) = \{T^{(k-1)}(w) \mid w \in N(v)\}$ as children.

Unfolding trees are a convenient tool to study the expressivity of GNNs as they are closely related to the 1-WL colors.

Lemma 3.2 ([12]). *Let $k \geq 0$ and u, v nodes, then $c_l^{(k)}(u) = c_l^{(k)}(v) \iff T^{(k)}(u) \simeq T^{(k)}(v)$.*

Xu et al. [68] show that GIN can distinguish nodes with different WL colors. The result is obtained by arguing that the MLP in Equation (4) is a universal function approximator [23] allowing to learn arbitrary permutation invariant functions [73]. This includes, in particular, injective functions. The complexity of GNNs regarding depth, width, and numerical precision required for this, however, remains poorly understood and is a focus of recent research [1].

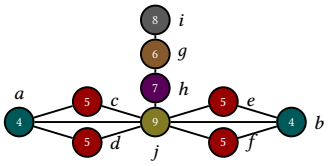
We investigate the functions of a GIN layer and their contribution to the expressivity of the final output, guiding the formulation of effective attacks. This requires determining whether our focus is on the expressivity of the general function on nodes or graphs in inductive learning or distinguishing the elements of a predefined subset in transductive settings. First, we consider the general case, where a finite-depth GNN processes all possible finite graphs and then discuss its implication for a concrete graph dataset. For the simplicity, we limit the discussion to unlabeled graphs.

Let $f^{(i)}\colon \mathcal{M}(\mathbb{R}^{d_{i-1}}) \to \mathbb{R}^{d_i}$ be the learnable function of the ith layer of a GNN, where $\mathcal{M}(U)$ are all possible pairs $(A, \mathcal{A})$ with $A \in U$ and $\mathcal{A}$ a countable multisets of elements from U. We assume that $f^{(i)}$ is invariant regarding the order of elements in the multiset $\mathcal{A}$. Then the output of the network for node v is obtained by the recursive function

$$F^{(k)}(v) = f^{(k)}\left(F^{(k-1)}(v), \{\!\{F^{(k-1)}(w) \mid w \in N(v)\}\!\}\right) \tag{8}$$

with $F^{(0)}$ uniform initial node features. Clearly, if all $f^{(i)}$ are injective, WL expressivity is reached as argued by Xu et al. [68]. The following proposition (proof in appendix) makes explicit that it suffices that all $f^{(i)}$ are injective with respect to the elements of their domain that represent (combinations of) unfolding trees of height $i - 1$.

Proposition 3.3. *Consider two arbitrary nodes u and v in an unlabeled graph. Let $\mathcal{J}_0$ be a uniform node feature and $\mathcal{J}_i = \{f^{(i)}(x) \mid$*



(a) Graph with final node coloring/embedding

	a	b	c	d	e	f	g	h	i	j
$F^{(0)}$	0	0	0	0	0	0	0	0	0	0
$F^{(1)}$	1	1	2	2	2	2	2	2	3	4
$\hat{F}^{(1)}$	2	2	2	2	2	2	2	2	2	3
$F^{(2)} = \hat{F}^{(2)}$	4	4	5	5	5	5	6	7	8	9

(b) Node colorings/embeddings at different layers

Figure 2: Exemplary results of the 2-layer GNNs $F^{(2)}$ and $\hat{F}^{(2)}$ using $f^{(i)}$ and $\hat{f}^{(i)}$, respectively, for $i \in \{1, 2\}$. Nodes having the same embedding are shown in the same color and are labeled with the same integer. Although $\hat{f}^{(1)}$ is non-injective and $\hat{F}^{(1)}$ is coarser than $F^{(1)}$, we have $F^{(2)} = \hat{F}^{(2)}$. The final output corresponds to the WL coloring.

$x \in \mathcal{M}(\mathcal{J}_{i-1})\}$ *the image under* $f^{(i)}$ *for* $i > 0$. *Then*

$$\forall i \leq k\colon \forall x, y \in \mathcal{M}(\mathcal{J}_{i-1})\colon f^{(i)}(x) = f^{(i)}(y) \implies x = y \tag{9}$$

implies

$$c_l^{(k)}(u) = c_l^{(k)}(v) \iff F^{(k)}(u) = F^{(k)}(v). \tag{10}$$

This result extends to graphs with discrete labels and continuous attributes. The inputs, requiring distinct outputs from a GIN layer to achieve WL expressivity, indicate weak points for potential attacks. These inputs are in 1-to-1 correspondence with the unfolding trees. As i increases, the number of unfolding trees and the discriminative complexity for GIN's MLPs grows. However, Proposition 3.3 provides only a sufficient condition for WL expressivity. Specifically, in a transductive setting on a concrete dataset, the variety of unfolding trees is limited by the dataset's node count. Additionally, even for a concrete dataset, the function $f^{(i)}$ at layer i might be non-injective while $F^{(i+1)}$ remains maximally expressive, as illustrated in Figure 2b. This motivates the need for a targeted attack on injectivity to effectively degrade expressivity, which we develop below. Further, these considerations lead to the following exemplary classification tasks.

3.2 Weisfeiler-Leman Discrimination Tasks

To develop an effective attack, we first need to provide a formulation of Prop. 3.3 that is more suitable to practical machine learning applications. Thus, we first introduce the following exemplary node level classification task, which we then extend to the graph level classification tasks for our experimental evaluation.

Definition 3.4 (WL-discrimination task). Let G be a graph with labels l. The *WL-discrimination task* for k in $\mathbb{N}$ is to learn a function $F^{(k)}$ such that $F^{(k)}(u) = F^{(k)}(v) \iff c_l^{(k)}(u) = c_l^{(k)}(v)$ for all $u, v \in V(G)$.

The definition, which in its above form is concerned solely with node classification based on specific structural features, can easily be extended to classifying graphs based on their structure.

To define a graph level WL discrimination task, we first extend Equation (8) with a readout function $f_G\colon \mathcal{A} \to \mathbb{R}^{d_G}$, mapping a countable multiset $\mathcal{A}$ of elements from U to a real valued vector representation of G:

$$F_G^{(k)}(G) = f_G(\{\!\{F^{(k)}(v) \mid v \in V(G)\}\!\})$$

Obviously, a GNN computing such a function is maximally expressive if $F_G^{(k)}(G_i) = F_G^{(k)}(G_j) \iff C_l^{(k)}(G_i) = C_l^{(k)}(G_j)$ for all G_i, G_j. It further follows directly from $C_l^{(k)}(G) = \{\!\{c_l^{(k)}(v) \mid v \in V(G)\}\!\}$ that if the kth layer's message passing computation is maximally expressive, $c_l^{(k)}(u) = c_l^{(k)}(v) \iff F^{(k)}(u) = F^{(k)}(v)$, and f_G's injectivity is a sufficient condition for $F_G^{(k)}(G_i) = F_G^{(k)}(G_j) \iff C_l^{(k)}(G_i) = C_l^{(k)}(G_j)$ to hold (Proposition 3.5, proof in appendix), which is consistent with results by Xu et al. [68].

Proposition 3.5. *Consider two arbitrary graphs* G_i, G_j, *with* $\mathcal{A} = \{\!\{F^{(k)}(v) \mid v \in V(G_i)\}\!\}$, $\mathcal{B} = \{\!\{F^{(k)}(v) \mid v \in V(G_j)\}\!\}$ *and* $c_l^{(k)}(u) = c_l^{(k)}(v) \iff F^{(k)}(u) = F^{(k)}(v)$. *Then*

$$f_G(\mathcal{A}) = f_G(\mathcal{B}) \implies \mathcal{A} = \mathcal{B} \tag{11}$$

implies

$$F_G^{(k)}(G_i) = F_G^{(k)}(G_j) \iff C_l^{(k)}(G_i) = C_l^{(k)}(G_j) \tag{12}$$

Note that analogous to Proposition 3.3, the injectivity of f_G with respect to the elements of its domain is only sufficient and not necessary for WL expressivity. In particular, the sets $\mathcal{A}$ and $\mathcal{B}$ have a specific structure as they represent the unfolding trees of graphs. Hence, non-injective realization of f_G might still suffice to distinguish graphs according to Equation (12).

Definition 3.6 (GLWL-discrimination task). Let D be a set of node labeled graphs and let $G_i, G_j \in D$. The *Graph Level WL-discrimination task* for k in $\mathbb{N}$ is to learn a function $F_G^{(k)}$ such that $F_G^{(k)}(G_i) = F_G^{(k)}(G_j)$ if and only if $C_l^{(k)}(G_i) = C_l^{(k)}(G_j)$ for all $G_i, G_j \in D$ after k iterations of the WL algorithm.

According to Propositions 3.3 and 3.5, the injectivity of $f^{(i)}$ and f_G is sufficient for achieving WL-level expressivity. Therefore, if a GNN has successfully learned a function $F_G^{(k)}$ for a GLWL discrimination task, any BFA targeting the expressivity of such a GNN will likely compromise the injectivity of either f_G or $f^{(i)}$.

Real-world graph datasets, however, rarely satisfy the strict conditions outlined in Definition 3.4 and 3.6 as classification targets used for supervised training typically do not perfectly align with the node's WL colors or the isomorphism types of the dataset. Consequently, we formulate a relaxation of Definition 3.6 based on the Jaccard distance for multisets.

Definition 3.7 (Multiset Jaccard distance [46]). Let A, B be multisets in universe U and m be the multiplicity function of multisets. The *multiset Jaccard distance* is then defined as $J_m(A, B) =$

$1 - \frac{\sum_{x \in U} \min(m_A(x), m_B(x))}{\sum_{x \in U} \max(m_A(x), m_B(x))}$, where $m_A(x)$ and $m_B(x)$ represent the multiplicity of element x in multisets A and B, respectively.

Definition 3.8 (ε-GLWL-discrimination task). Let $D = \bigcup_{c \in C} D_c$ be a set of node labeled graphs with class labels C, where D_c are the graphs of class $c \in C$. The *ε-GLWL-discrimination task* for k in $\mathbb{N}$ is to learn a function $F_G^{(k)}$ such that for all $c \in C$ and all G_i, G_j in $D_c = \{G_1, G_2, \ldots, G_n\}$ it holds that $F_G^{(k)}(G_i) = F_G^{(k)}(G_j)$ if and only if $S_c \leq \varepsilon$ with $S_c = \frac{1}{\delta_c} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} J_m(C_l^{(k)}(G_i), C_l^{(k)}(G_j))$ for some $\varepsilon \in \mathbb{R}, 0 \leq \varepsilon < 1$ after k iterations of the WL algorithm and $\delta_c = \binom{|D_c|}{2}$ denoting the number of unique pairs in D_c.

Definition 3.8 captures a broader trend regarding structural discrimination in the learning task, as it permits some graphs with the same classification target to exhibit a degree (specified by ε) of structural dissimilarity. Likewise, it permits graphs with different targets to show a degree of structural similarity. Nonetheless, any $\varepsilon < 1$ still requires $F_G^{(k)}$ to learn to distinguish certain substructures, for which injectivity of $f^{(k)}$ as well as f_G with respect to an accordingly constrained domain remains a sufficient condition. Note that Definition 3.8 provides per-class formulation. We use a per-dataset extension $\frac{1}{|C|} \sum_{c \in C} S_c \leq \varepsilon$ in Figure 5 for ease of display.

3.3 Targeting injectivity

It would not suffice to consider the injectivity of a single layer's Combine and Aggregate functions for an attack, as expressivity could be restored at deeper layers (Section 3.1, Figure 2b). Thus, to target the injectivity sufficent for (ε-GL)WL-discrimination tasks as per Definition 3.4- 3.8 while considering the entire model, we reformulate the target of PBFA from a maximization Equation (7) to a minimization problem

$$\min_{\{\widehat{\mathbf{W}}_q^{(l)}\}} \mathcal{L}\Big(\Phi(\mathbf{X}_a; \{\widehat{\mathbf{W}}_q^{(l)}\}_{l=1}^{L}),\ \Phi(\mathbf{X}_b; \{\widehat{\mathbf{W}}_q^{(l)}\}_{l=1}^{L})\Big). \tag{13}$$

That is, instead of increasing, e.g., the original classification loss of the model Φ via PBS, we use PBS to minimize the difference between the outputs of the network computed on two different inputs $\mathbf{X}_a$, and $\mathbf{X}_b$ w.r.t. the function $\mathcal{L}$ that measures the difference between the network's outputs. This connects with our theoretical framework via (ε-GL)WL-discrimination tasks, in which classification targets (tend to) correlate with graph structural (dis)similarity. Thus, inducing bit flips that exploit this tendency will potentially impair the injective mappings facilitated by the GNNs' Aggregate and Combine functions. Naturally, such an attack strategy necessaites careful selection of loss function and input data samples, which we discuss in detail below. This approach further allows us to perform IBFA on unlabeled data.

3.3.1 Choosing the loss function. In a binary graph classification task, the network's outputs $\mathbf{y}_a = \Phi(\mathbf{X}_a; \{\widehat{\mathbf{W}}_q^{(l)}\}_{l=1}^{L})$ as well as $\mathbf{y}_b = \Phi(\mathbf{X}_b; \{\widehat{\mathbf{W}}_q^{(l)}\}_{l=1}^{L})$ both are $n \times 1$ vectors representing the probability mass functions (PMF) of n Bernoulli distributed discrete random variables. For such distributed output vectors, any differentiable p-norm-based loss function would suffice to converge predictions in the sense of Equation (13) and we choose L1 for $\mathcal{L}$ for simplicity. In non-binary graph classification (i.e., multiclass-classification) or multitask binary classification, however, outputs $\mathbf{Y}_a$ and $\mathbf{Y}_b$ are not $n \times 1$ vectors but instead $n \times m$ matrices where n is the number of samples and m the number of classes/tasks. That is, for each of the n samples, each column in $\mathbf{Y}_a$ and $\mathbf{Y}_b$ represents a PMF over m classes. Thus, simply using a p-norm-based loss function as L1 for $\mathcal{L}$ in (13) would fail to capture differences in individual class probabilities contained in $\mathbf{Y}_a$ and $\mathbf{Y}_b$ due to the reduction operation required by L1 (e.g., mean or sum over m). We solve this by, instead of L1, employing the discrete pointwise Kullback-Leibler-Divergence [30] (KL) as $\mathcal{L}$, i.e., the KL between the output's n probability distributions of each pair of samples (data points) in $\mathbf{Y}_a$ and $\mathbf{Y}_b$, which, in the context of (13), allows IBFA to find bits converging the PMF of $\mathbf{Y}_a$ best on $\mathbf{Y}_b$.

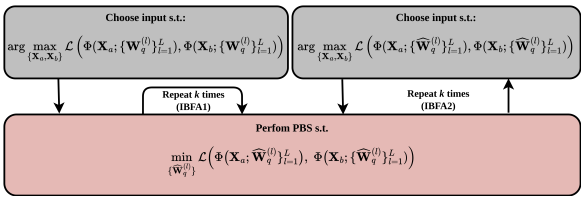


Figure 3: IBFA1/2's integration of PBS and input data selection strategies for k attack iterations. In the first attack iteration, input data selection of IBFA1 and IBFA2 are identical.

3.3.2 Choosing input samples. The proper selection of $\mathbf{X}_a$ and $\mathbf{X}_b$ is crucial as selecting inputs that have identical outputs (e.g., two batches that contain different samples of the same classes in the same order) before the attack will not yield any degradation as Equation (13) would already be optimal. Thus we chose inputs $\mathbf{X}_a$ and $\mathbf{X}_b$ to be maximally different from one another w.r.t. to the unperturbed network's outputs by

$$\arg\max_{\{\mathbf{X}_a,\mathbf{X}_b\}} \mathcal{L}\Big(\Phi(\mathbf{X}_a; \{\mathbf{W}_q^{(l)}\}_{l=1}^{L}),\ \Phi(\mathbf{X}_b; \{\mathbf{W}_q^{(l)}\}_{l=1}^{L})\Big). \tag{14}$$

This search mechanism can be executed before the attack and the found $\mathbf{X}_a, \mathbf{X}_b$ reused for all attack iterations, a variant of IBFA to which we refer as **IBFA1**. However, after each attack iteration, the solution of (14) might change, making it promising to recompute $\mathbf{X}_a$, $\mathbf{X}_b$ on the perturbed model before every subsequent attack iteration. We refer to the IBFA employing the latter data selection strategy as **IBFA2**. While IBFA2 may lead to slightly faster and more consistent degradation for a set amount of bit flips, its time complexity of $\Theta(kn^2)$ for k attack runs and n samples makes it less suitable for large datasets. An overview of IBFA1 and IBFA2 is provided in Figure 3.

3.4 Assumptions and threat model

Our work builds on several fundamental assumptions which are in line with previous work on BFA-based attacks on various types of neural networks. Following previous literature on BFAs for CNNs, we assume our target network is `INT8` quantized [36, 48, 49, 71], as such configured networks are naturally noise resistant [49].

Furthermore, we adopt the usual premise that the attacker has the capability to exactly flip the bits chosen by the bit-search algorithm through mechanisms such as RowHammer [45], variants

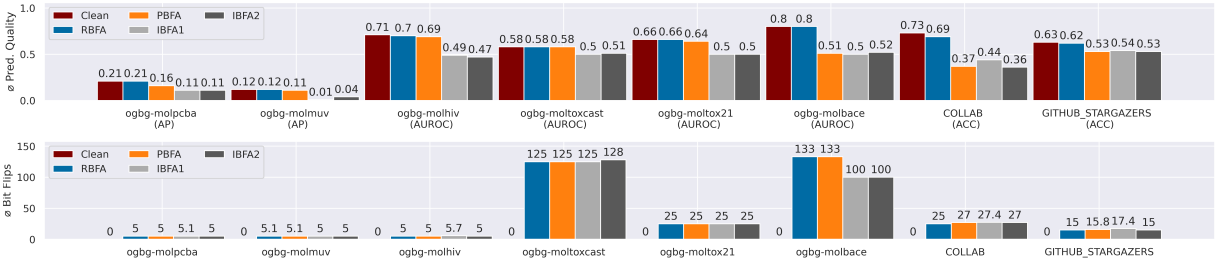


Figure 4: Pre- (clean) and post-attack test quality metrics AP, AUROC or ACC for different BFA variants on a 5-layer GIN trained on six `ogbg-mol` and two TUDataset datasets, number of bit flips, averages of 10 runs.

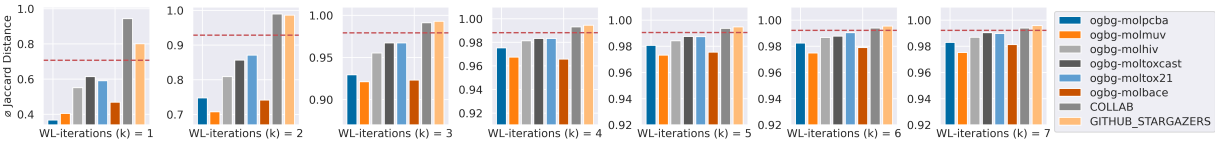


Figure 5: Jaccard distances $J_m(C_l^{(k)}(G_i), C_l^{(k)}(G_j)), i \neq j$, averaged over all graphs G_i, G_j and tasks (classes) in each randomly drawn sample of 3200 graphs per dataset, indicating ε-GLWL-discrimination tasks. The dotted lines distinguish molecular and social datasets for various numbers of WL-iterations (k) from 1 to 7 and suggest possible choices for ε.

thereof [35, 71] or others [7, 24]. The feasibility of BFAs inducing exact bit flips via a RowHammer variant was shown by, e.g., [60]. We thus do not discuss the detailed technical specialities of realizing the flips of the identified vulnerable bits in hardware.

Moreover, we assume a gray-box attack model in which the attacker's goal is to crush a trained and deployed quantized GNN via BFA. It represents a relaxation of some of the restrictions found in a black-box model by allowing the attacker to have certain prior knowledge about the training data and the general structure of the model. This is again a typical assumption also made in previous work [36]. An attacker could easily obtain required information in many typical application scenarios, for instance, if the target model is a publicly available pre-trained model or the attacker has access to an identical instance of a device on which model inference is performed. Even in the absence of such a priori knowledge on the model, parameters and input data might be acquired through methods such as side-channel attacks [4, 69].

4 EXPERIMENTS

Our experimental framework is designed to investigate the following key research questions, focusing on real-world molecular property prediction datasets, a task common in drug development [51, 65], as well as social network classification.

RQ1 Is IBFA more destructive than other BFAs?
RQ2 Does the destructiveness of IBFA depend on whether a task requires high structural expressivity (i.e., is an ε-GLWL-discrimination task)?
RQ3 Does IBFA's required number of bit flips fall within the accepted realistic budget of 24 to 50, as suggested in related work [60, 71]?
RQ4 How does IBFA perform in scenarios with limited data available for selection?

We assess IBFA relative to PBFA, which we consider the most relevant baseline, as most other, more specialized (e.g., targeted) BFAs designed to degrade CNNs have been derived from PBFA. We measured the degradation in the quality metrics proposed by Open Graph Benchmark (OGB) [25] or TUDataset [42], respectively, for each of the datasets and followed the recommended variant of a 5-layers GIN with a virtual node. For thoroughness, we include a detailed ablation study concerning loss functions and layer preferences, which can be found in the appendix. This study also covers experiments on GNNs less expressive than GIN and IBFA's relation to certain BFA defenses. To ensure reproducibility, we provide details on quantized models, measured metrics and attack configuration and a code repository[1].

4.1 Quantized models

We obtained `INT8` quantized models by training on each dataset's training split using STE, see Section 2.5. We used the Adam optimizer with a learning rate of 10^{-3} and trained the models for 30 epochs. Although more complex models and quantization techniques might achieve higher prediction quality, our focus was not on improving prediction quality beyond the state-of-the-art, but on demonstrating GIN's vulnerability to IBFA. Some of the datasets we used present highly challenging learning tasks, and our results

[1] https://github.com/lorenz0890/ibfakdd2024

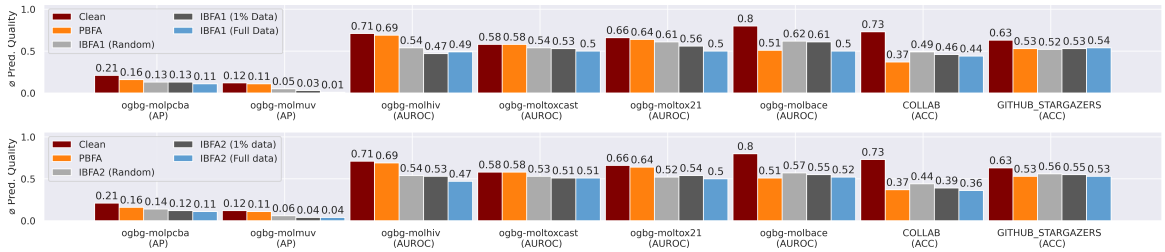


Figure 6: Pre- (clean) and post-attack test quality metrics AP, AUROC or ACC for IBFA with different data selection strategies (random, IBFA selection from 1% random subsets and full training splits) on a 5-layer GIN trained on 6 `ogbg-mol` and 2 TUDataset datasets, averages of 10 runs. IBFA1 in top row, IBFA2 in bottom row, number of bit flips as reported in Figure 4.

for quantized training of GIN are comparable to those by OGB [25] for `FLOAT32` training.

4.2 Datasets

Six benchmark datasets are chosen (as in, e.g., [15, 57]) from graph classification tasks from OGB based on MoleculeNet [64] for evaluation as well as `COLLAB` and `GITHUB_STARGAZERS` from TUDataset [42]. The goal in each OGB dataset is to predict properties based on molecular graph structures, such that the datasets are consistent with the underlying assumptions of IBFA described in Section 3. All OGB datasets are split using a scaffold-based strategy, which seeks to separate structurally different molecules into different subsets [25, 64]. `COLLAB` is derived from scientific collaboration networks, whereby every graph represents the ego-network of a scientist, and the task is to predict their area of research. `GITHUB_STARGAZERS` contains graphs of social networks of GitHub users, and the task is to predict whether they starred popular machine learning or web development repositories. `COLLAB` and `GITHUB_STARGAZERS` are split randomly (80/10/10 for train/test/validation).

Not all targets in the OGB datasets apply to each molecule (missing targets are indicated by NaNs) and we consider only existing targets in our experiments. Area under the receiver operating characteristic curve (AUROC), average precision (AP) or accuracy (ACC) are used to measure the models' performance as recommended by Hu et al. [25] and Morris et al. [42], respectively.

4.3 Attack configuration

The attacks in each of the experiments on a GIN trained on a dataset were executed with the number of attack runs (see Section 2.6) initially set to 5 and repeated with the number of attacks incremented until the first attack type reached (nearly) random output. The other attacks in this experiment were then set to that same number of attack runs to ensure fair comparison. Note that PBFA, IBFA1 and IBFA2 can flip more than a single bit during one attack run (see Section 2.6), such that the final number of actual bit flips can vary across experiments. For the single task binary classification datasets, `ogbg-molhiv`, `ogbg-bace` and `GITHUB_STARGAZERS`, IBFA1/2 were used with $L1$ loss, for multitask binary classification `ogbg-tox21`, `ogbg-toxcast`, `ogbg-molmuv`, `ogbg-pcba` and multiclass classification (`COLLAB`), IBFA1/2 were used with KL loss. For PBFA, binary CE (BCE) loss was used throughout the binary classification datasets

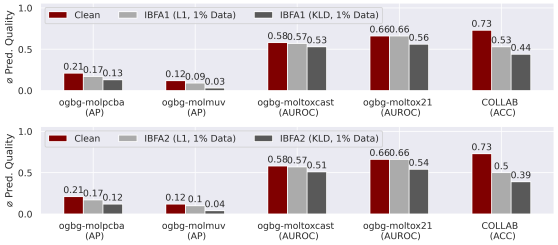


Figure 7: Pre- (clean) and post-attack test quality metrics AP, AUROC or ACC for IBFA with different loss functions (L1 vs. KL loss, IBFA selection from 1% random subset) on a 5-layer GIN trained on 4 `ogbg-mol` datasets TUDataset `COLLAB`, number of bit flips, averages of 10 runs. IBFA1 in top row, IBFA2 in bottom row.

and CE loss was used for `COLLAB`. Input samples for all evaluated BFA variants were taken from the training splits.

4.4 Results

Both IBFA variants surpass random bit flips (**RBFA**) and PBFA in terms of test quality metric degradation for a given number of bit flips in most examined cases (Figure 4) (**RQ1**). A fine granular visualization of the progression of quality metric degradation induced by IBFA1/2, PBFA and RBFA is given in Figure. 8. IBFA is capable of forcing the evaluated GINs to produce (almost) random output (AUROC $\leq$ 0.5, AP $\leq$ 0.11 (`ogbg-molpcba`), AP $\leq$ 0.06 (`ogbg-molmuv`), ACC $\leq$ 0.33 (`COLLAB`) or $\leq$ 0.5 (`GITHUB_STARGAZERS`)) by flipping less than 33 bits on average. This is realizable given the aforementioned upper limit of 24 to 50 precise bit flips an attacker can be expected to achieve [60, 71] (**RQ3**). IBFA2 causes slightly more quality metric degradation on `ogbg-molhiv`, `COLLAB` and `GITHUB_STARGAZERS` than IBFA1 but is surpassed by or on par with IBFA1 in all other cases. IBFA2 on `ogbg-molbace` and IBFA1 on `COLLAB` were slightly weaker than PBFA. However, on `ogbg-molbace` PBFA requires 33% more bit flips to achieve results compareable to IBFA1. On `GITHUB_STARGAZERS`, PBFA and IBFA both degraded GIN equally. GINs trained on `ogbg-molmuv`, `ogbg-molhiv`, and `ogbg-moltox21` were barely affected by PBFA for the examined number of bit flips and GIN trained on `ogbg-moltoxcast`

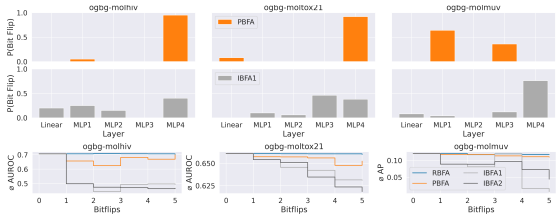


Figure 8: Probability of selecting a layer for a bit flip in a 5-layer GIN trained on 3 small datasets, averaged over 10 runs for PBFA or IBFA1 (top row), progression of degradation of for different BFA variants (bottom row).

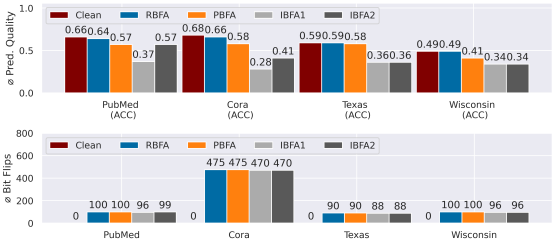


Figure 9: Pre- (clean) and post-attack test quality metric ACC for different BFA variants on a 5-layer GIN trained on four TUDataset node level datasets, number of bit flips, averages of 10 runs/folds.

appeared to be entirely impervious to PBFA. Our quantized GNNs resist RBFA, ruling out our observations are stochastic.

These results are consistent with our definition of ε-GLWL-discrimination tasks as Figure 5 indicates that, for suiteable choices of ε, the examined molecular graph property prediction datasets from OGB could be distinguished from the social network classification datasets from TUDatset. That is, the OGB datasets demonstrate a stronger connection between class membership and structural graph similarity and thus, GINs trained on them are more vulnerable to IBFA. Together with our main results (Figure 4), which show that PBFA either causes less degradation in the GINs trained on the OGB datasets than IBFA or requires more flips to do so, we interpret these findings as empirical validation of our theoretical prediction that IBFA's destructiveness surpasses that of PBFA in tasks demanding high structural expressivity (**RQ2**). The proposed data selection strategies for IBFA1/2 provide an improvement over random data selection (Figure 6) and IBFA1/2 remain highly destructive (typically more destructive than PBFA), even when limited to a significantly constrained sample (1% random subsets of the datasets' training splits) for selecting input data points (**RQ4**).

While our setup did not permit the weaker BFA in an experiment to continue flipping bits until the network was fully degraded, our preliminary case study (appendix) revealed PBFA required 953 bit flips to fully degrade GIN on `ogbg-molhiv` and failed to degrade GIN on `ogbg-moltoxcast` even after 2662 bit flips. For comparison, IBFA1/2 could fully degrade GIN on both `ogbg-molhiv` and `ogbg-moltoxcast` using two orders of magnitude fewer bit flips (Figure 4).

Ablation experiments, layer preferences, node classification. Figure 7 illustrates the results if L1 loss is used instead of KL loss in the multi-task binary classification setting. As can be seen from Figure 7, IBFA1/2 both fail to outperform PBFA on the multi-task binary classification datasets if L1 loss is used instead of KL loss, which is in line with our analytical results in Section 3.3.1. As in Figure 6, 1% subset sampling was used for IBFA to accelerate the experiments. Experiments on `obg-molbace`, `ogbg-molhiv` and `GITHUB_STARGAZERS` are not included in this experiment as we already used L1 loss in our original experiments with these datasets.

We recorded the probabilities associated with an attack's selection of a specific layer within the evaluated 5-layer GIN (Figure 8). RBFA was configured to exhibit a random uniform distribution of bit flips across layers and is therefore omitted. PBFA and IBFA1 exhibit a distinct and characteristic layer selection. PBFA typically confines bit flips to only 2 out of the 5 layers and, in line with Hector et al. [20]'s findings for CNNs, displays a preference for the input layer, while IBFA1/2 targets at least 4 layers across the entire model, with the majority of flips occurring in the learnable aggregation functions of the network (MLP1-4). The variations in layer selection observed in IBFA1 support our hypotheses: a) introducing non-injectivity into a single layer alone is insufficient, necessitating an attack on the overall expressivity of GIN, and b) IBFA1 effectively targets the learnable neighborhood aggregation functions.

Beyond graph classification, we experimented with two heterophilious [19] node classification datasets, `Texas` and `Wisconsin` from TUDataset, using the same GIN architecture as in our graph-level experiments, sans readout. Results, averaged over 10 predefined folds (Figure 9), show that IBFA1/2 heavily degrade performance while PBFA is nearly indistinguishable from RBFA. We also tested our method on homophilious [19] node classification datasets, `Cora` and `PubMed`, reporting 10-run averages. Here, we observed a different trend: while IBFA1 remained effective, PBFA degraded accuracy more than RBFA, particularly on `Cora`. These results suggest that PBFA's reliance on pseudo labels in its loss function makes it less effective when these labels are poor, whereas IBFA's effectiveness remains robust regardless of label quality.

5 CONCLUSION

We introduce a novel concept for bit flip attacks on quantized GNNs which exploits specific mathematical properties of GNNs related to graph learning tasks requiring graph structural discrimination. Upon our theory, we design the novel Injective Bit Flip Attack IBFA and illustrate its ability to render GIN indifferent to graph structures. IBFA compromises its predictive quality significantly more than the most relevant BFA ported from CNNs and than random bit flips on twelve different datasets, covering binary and multiclass classification of molecular, social and other graphs or nodes.

ACKNOWLEDGEMENTS

This work was supported by the Vienna Science and Technology Fund (WWTF) [10.47379/VRG19009], the Austrian Federal Ministry of Labour and Economy, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association. We thank Tabea Reichmann for useful discussions.

APPENDIX A.1 • ATTACKING GRAPH NEURAL NETWORKS WITH BIT FLIPS

REFERENCES

[1] Anders Aamand, Justin Chen, Piotr Indyk, Shyam Narayanan, Ronitt Rubinfeld, Nicholas Schiefer, Sandeep Silwal, and Tal Wagner. 2022. Exponentially Improving the Complexity of Simulating the Weisfeiler-Lehman Test with Graph Neural Networks. In *Advances in Neural Information Processing Systems 35*. 27333–27346.

[2] Mohammad-Hossein Askari-Hemmat, Sina Honari, Lucas Rouhier, Christian S. Perone, Julien Cohen-Adad, Yvon Savaria, and Jean-Pierre David. 2019. U-net fixed-point quantization for medical image segmentation. In *Large-Scale Annotation of Biomedical Data and Expert Label Synthesis (LABELS) and Hardware Aware Learning for Medical Imaging and Computer Assisted Intervention (HAL-MICCAI), International Workshops*. 115–124.

[3] Mehdi Bahri, Gaétan Bahl, and Stefanos Zafeiriou. 2021. Binary graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9492–9501.

[4] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. 2018. CSI neural network: Using side-channels to recover your artificial neural network information. *CoRR* abs/2204.07697 (2018).

[5] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *CoRR* abs/1308.3432 (2013).

[6] Blaž Bertalanič and Carolina Fortuna. 2023. Graph Isomorphism Networks for Wireless Link Layer Anomaly Classification. In *2023 IEEE Wireless Communications and Networking Conference (WCNC)*. 1–6.

[7] Jakub Breier, Xiaolu Hou, Dirmanto Jap, Lei Ma, Shivam Bhasin, and Yang Liu. 2018. Practical Fault Attack on Deep Neural Networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2204–2206.

[8] Mark Cheung and José MF Moura. 2020. Graph neural networks for covid-19 drug discovery. In *2020 IEEE International Conference on Big Data*. 5646–5648.

[9] Enyan Dai, Tianxiang Zhao, Huaisheng Zhu, Junjie Xu, Zhimeng Guo, Hui Liu, Jiliang Tang, and Suhang Wang. 2022. A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability. *CoRR* abs/2204.08570 (2022).

[10] Austin Derrow-Pinion, Jennifer She, David Wong, Oliver Lange, Todd Hester, Luis Perez, Marc Nunkesser, Seongjae Lee, Xueying Guo, Brett Wiltshire, et al. 2021. Eta prediction with graph neural networks in google maps. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3767–3776.

[11] Guimin Dong, Mingyue Tang, Zhiyuan Wang, Jiechao Gao, Sikun Guo, Lihua Cai, Robert Gutierrez, Bradford Campbel, Laura E Barnes, and Mehdi Boukhechba. 2023. Graph neural networks in IoT: a survey. *ACM Transactions on Sensor Networks* 19, 2 (2023), 1–50.

[12] Giuseppe Alessio D'Inverno, Monica Bianchini, Maria Lucia Sampoli, and Franco Scarselli. 2021. A unifying point of view on expressive power of GNNs. *CoRR* abs/2106.08992 (2021).

[13] Boyuan Feng, Yuke Wang, Xu Li, Shu Yang, Xueqiao Peng, and Yufei Ding. 2020. SGQuant: Squeezing the Last Bit on Graph Neural Networks with Specialized Quantization. In *2020 IEEE 32nd international conference on tools with artificial intelligence (ICTAI)*. 1044–1052.

[14] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[15] Han Gao, Xu Han, Jiaoyang Huang, Jian-Xun Wang, and Liping Liu. 2022. PatchGT: Transformer over Non-trainable Clusters for Learning Graph Representations. In *Learning on Graphs Conference*. 1–27.

[16] Jianliang Gao, Tengfei Lyu, Fan Xiong, Jianxin Wang, Weimao Ke, and Zhao Li. 2022. Predicting the Survival of Cancer Patients With Multimodal Graph Neural Network. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 19, 2 (2022), 699–709.

[17] Yun Gao, Hirokazu Hasegawa, Yukiko Yamaguchi, and Hajime Shimada. 2022. Malware Detection by Control-Flow Graph Level Representation Learning With Graph Isomorphism Network. *IEEE Access* 10 (2022), 111830–111841.

[18] Mukhammed Garifulla, Juncheol Shin, Chanho Kim, Won Hwa Kim, Hye Jung Kim, Jaeil Kim, and Seokin Hong. 2021. A case study of quantizing convolutional neural networks for fast disease diagnosis on portable medical devices. *Sensors* 22, 1 (2021), 219.

[19] Jhony H Giraldo, Konstantinos Skianis, Thierry Bouwmans, and Fragkiskos D. Malliaros. 2023. On the trade-off between over-smoothing and over-squashing in deep graph neural networks. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 566–576.

[20] Kevin Hector, Pierre-Alain Moëllic, Mathieu Dumont, and Jean-Max Dutertre. 2022. A Closer Look at Evaluating the Bit-Flip Attack Against Deep Neural Networks. In *2022 IEEE 28th International Symposium on On-Line Testing and Robust System Design*. 1–5.

[21] Sanghyun Hong, Pietro Frigo, Yiğitcan Kaya, Cristiano Giuffrida, and Tudor Dumitraş. 2019. Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks. In *28th USENIX Security Symposium (USENIX Security 19)*. 497–514.

[22] Kurt Hornik. 1991. Approximation capabilities of multilayer feedforward networks. *Neural networks* 4, 2 (1991), 251–257.

[23] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2, 5 (1989), 359–366.

[24] Xiaolu Hou, Jakub Breier, Dirmanto Jap, Lei Ma, Shivam Bhasin, and Yang Liu. 2020. Security evaluation of deep neural network resistance against laser fault injection. In *2020 IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits*. 1–6.

[25] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in neural information processing systems 33*. 22118–22133.

[26] Stefanie Jegelka. 2022. Theory of graph neural networks: Representation and learning. In *Proceedings of the International Congress of Mathematicians*, Vol. 7. 5450–5476.

[27] Xun Jiao, Ruixuan Wang, Fred Lin, Daniel Moore, and Sriram Sankar. 2022. PyGFI: Analyzing and Enhancing Robustness of Graph Neural Networks Against Hardware Errors. *CoRR* abs/2212.03475 (2022).

[28] Wei Jin, Yaxing Li, Han Xu, Yiqi Wang, Shuiwang Ji, Charu Aggarwal, and Jiliang Tang. 2021. Adversarial attacks and defenses on graphs. *ACM SIGKDD Explorations Newsletter* 22, 2 (2021), 19–34.

[29] Yash Khare, Kumud Lakara, Maruthi S. Inukonda, Sparsh Mittal, Mahesh Chandra, and Arvind Kaushik. 2022. Design and Analysis of Novel Bit-flip Attacks and Defense Strategies for DNNs. In *2022 IEEE Conference on Dependable and Secure Computing*. 1–8.

[30] Solomon Kullback and Richard Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22 (1951), 79–86.

[31] Lorenz Kummer, Kevin Sidak, Tabea Reichmann, and Wilfried Gansterer. 2023. Adaptive Precision Training (AdaPT): A dynamic quantized training approach for DNNs. In *Proceedings of the 2023 SIAM International Conference on Data Mining*. 559–567.

[32] Jingtao Li, Adnan Siraj Rakin, Zhezhi He, Deliang Fan, and Chaitali Chakrabarti. 2021. RADAR: Run-time Adversarial Weight Attack Detection and Accuracy Recovery. In *2021 Design, Automation and Test in Europe Conference and Exhibition*. 790–795.

[33] Yang Li, Buyue Qian, Xianli Zhang, and Hui Liu. 2020. Graph neural network-based diagnosis prediction. *Big Data* 8, 5 (2020), 379–390.

[34] Xuan Lin, Zhe Quan, Zhi-Jie Wang, Tengfei Ma, and Xiangxiang Zeng. 2020. KGNN: Knowledge Graph Neural Network for Drug-Drug Interaction Prediction. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, Vol. 380. 2739–2745.

[35] Moritz Lipp, Michael Schwarz, Lukas Raab, Lukas Lamster, Misiker Tadesse Aga, Clémentine Maurice, and Daniel Gruss. 2020. Nethammer: Inducing rowhammer faults through network requests. In *2020 IEEE European Symposium on Security and Privacy Workshops*. 710–719.

[36] Qi Liu, Jieming Yin, Wujie Wen, Chengmo Yang, and Shi Sha. 2023. NeuroPots: Realtime Proactive Defense against Bit-Flip Attacks in Neural Networks. In *32nd USENIX Security Symposium (USENIX Security 23)*. 6347–6364.

[37] Yannan Liu, Lingxiao Wei, Bo Luo, and Qiang Xu. 2017. Fault injection attack on deep neural network. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 131–138.

[38] Zheng Liu, Xiaohan Li, Hao Peng, Lifang He, and S Yu Philip. 2020. Heterogeneous similarity graph neural network on electronic health records. In *2020 IEEE International Conference on Big Data*. 1196–1205.

[39] Haohui Lu and Shahadat Uddin. 2021. A weighted patient network-based framework for predicting chronic diseases using graph neural networks. *Scientific reports* 11, 1 (2021), 22607.

[40] Jiaqi Ma, Shuangrui Ding, and Qiaozhu Mei. 2020. Towards More Practical Adversarial Attacks on Graph Neural Networks. In *Advances in Neural Information Processing Systems 33*. 4756–4766.

[41] Christopher Morris, Matthias Fey, and Nils Kriege. 2021. The Power of the Weisfeiler-Leman Algorithm for Machine Learning with Graphs. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. 4543–4550.

[42] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*.

[43] Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M. Kriege, Martin Grohe, Matthias Fey, and Karsten Borgwardt. 2023. Weisfeiler and Leman go Machine Learning: The Story so far. *Journal of Machine Learning Research* 24, 333 (2023), 1–59.

[44] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 4602–4609.

[45] Onur Mutlu and Jeremie S. Kim. 2019. Rowhammer: A retrospective. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 8 (2019), 1555–1571.
[46] Javier Parapar and Álvaro Barreiro. 2008. Winnowing-Based Text Clustering. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. 1353–1354.
[47] Cheng Qian, Ming Zhang, Yuanping Nie, Shuaibing Lu, and Huayang Cao. 2023. A Survey of Bit-Flip Attacks on Deep Neural Network and Corresponding Defense Methods. *Electronics* 12, 4 (2023), 853.
[48] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. 2019. Bit-Flip Attack: Crushing Neural Network With Progressive Bit Search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision and Pattern Recognition*. 1211–1220.
[49] Adnan Siraj Rakin, Zhezhi He, Jingtao Li, Fan Yao, Chaitali Chakrabarti, and Deliang Fan. 2022. T-BFA: Targeted Bit-Flip Adversarial Weight Attack. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 11 (2022), 7928–7939.
[50] Henrique De Melo Ribeiro, Ahran Arnold, James P. Howard, Matthew J. Shun-Shin, Ying Zhang, Darrel P. Francis, Phang B. Lim, Zachary Whinnett, and Massoud Zolgharni. 2022. ECG-based real-time arrhythmia monitoring using quantized deep neural networks: A feasibility study. *Computers in Biology and Medicine* 143 (2022), 105249.
[51] Ryan A Rossi, Di Jin, Sungchul Kim, Nesreen K. Ahmed, Danai Koutra, and John Boaz Lee. 2020. On proximity and structural role-based embeddings in networks: Misconceptions, techniques, and applications. *ACM Transactions on Knowledge Discovery from Data* 14, 5 (2020), 1–37.
[52] Till Hendrik Schulz, Tamás Horváth, Pascal Welke, and Stefan Wrobel. 2022. A Generalized Weisfeiler-Lehman Graph Kernel. *Machine Learning* 111, 7 (2022), 2601–2629.
[53] Yingxia Shao, Hongzheng Li, Xizhi Gu, Hongbo Yin, Yawen Li, Xupeng Miao, Wentao Zhang, Bin Cui, and Lei Chen. 2024. Distributed graph neural network training: A survey. *Comput. Surveys* 56, 8 (2024), 1–39.
[54] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research* 12, 9 (2011).
[55] Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. 2020. Adversarial Attacks on Graph Neural Networks via Node Injections: A Hierarchical Reinforcement Learning Approach. In *Proceedings of The Web Conference 2020*. 673–683.
[56] Zhenchao Sun, Hongzhi Yin, Hongxu Chen, Tong Chen, Lizhen Cui, and Fan Yang. 2021. Disease Prediction via Graph Neural Networks. *IEEE Journal of Biomedical and Health Informatics* 25, 3 (2021), 818–826.
[57] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. 2021. Adversarial graph augmentation to improve graph contrastive learning. In *Advances in Neural Information Processing Systems 34*. 15920–15933.
[58] Shyam A. Tailor, Javier Fernandez-Marques, and Nicholas D. Lane. 2021. Degree-Quant: Quantization-Aware Training for Graph Neural Networks. In *9th International Conference on Learning Representations*.
[59] Valerio Venceslai, Alberto Marchisio, Ihsen Alouani, Maurizio Martina, and Muhammad Shafique. 2020. Neuroattack: Undermining spiking neural networks security through externally triggered bit-flips. In *2020 International Joint Conference on Neural Networks*. 1–8.
[60] Jialai Wang, Ziyuan Zhang, Meiqi Wang, Han Qiu, Tianwei Zhang, Qi Li, Zongpeng Li, Tao Wei, and Chao Zhang. 2023. Aegis: Mitigating Targeted Bit-flip Attacks against Deep Neural Networks. In *32nd USENIX Security Symposium (USENIX Security 23)*. 2329–2346.
[61] Zhiqiong Wang, Zican Lin, Shuo Li, Yibo Wang, Weiying Zhong, Xinlei Wang, and Junchang Xin. 2023. Dynamic Multi-Task Graph Isomorphism Network for Classification of Alzheimer's Disease. *Applied Sciences* 13, 14 (2023), 8433.
[62] Bang Wu, Xingliang Yuan, Shuo Wang, Qi Li, Minhui Xue, and Shirui Pan. 2024. Securing Graph Neural Networks in MLaaS: A Comprehensive Realisation of Query-based Integrity Verification. In *2024 IEEE Symposium on Security and Privacy (SP)*. 110–110.
[63] Lingfei Wu, Peng Cui, Jian Pei, Liang Zhao, and Le Song. 2022. *Graph Neural Networks: Foundations, Frontiers, and Applications*.
[64] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. 2018. MoleculeNet: a benchmark for molecular machine learning. *Chemical science* 9, 2 (2018), 513–530.
[65] Jiacheng Xiong, Zhaoping Xiong, Kaixian Chen, Hualiang Jiang, and Mingyue Zheng. 2021. Graph neural networks for automated de novo drug design. *Drug Discovery Today* 26, 6 (2021), 1382–1393.
[66] Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. 2020. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing* 17 (2020), 151–178.
[67] Jingjing Xu, Wangchunshu Zhou, Zhiyi Fu, Hao Zhou, and Lei Li. 2021. A survey on green deep learning. *CoRR* abs/2111.05193 (2021).
[68] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *7th International Conference on Learning Representations*.
[69] Mengjia Yan, Christopher W Fletcher, and Josep Torrellas. 2020. Cache telepathy: Leveraging shared resource attacks to learn {DNN} architectures. In *29th USENIX Security Symposium (USENIX Security 20)*. 2003–2020.
[70] Sihong Yang, Dezhi Jin, Jun Liu, and Ye He. 2022. Identification of Young High-Functioning Autism Individuals Based on Functional Connectome Using Graph Isomorphism Network: A Pilot Study. *Brain Sciences* 12, 7 (2022), 883.
[71] Fan Yao, Adnan Siraj Rakin, and Deliang Fan. 2020. DeepHammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips. In *29th USENIX Security Symposium (USENIX Security 20)*. 1463–1480.
[72] Jiangchao Yao, Shengyu Zhang, Yang Yao, Feng Wang, Jianxin Ma, Jianwei Zhang, Yunfei Chu, Luo Ji, Kunyang Jia, Tao Shen, et al. 2022. Edge-cloud polarization and collaboration: A comprehensive survey for ai. *IEEE Transactions on Knowledge and Data Engineering* 35, 7 (2022), 6866–6886.
[73] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. 2017. Deep Sets. In *Advances in Neural Information Processing Systems 30*. 3391–3401.
[74] Rongzhao Zhang and Albert C. S. Chung. 2021. MedQ: Lossless ultra-low-bit neural network quantization for medical image segmentation. *Medical Image Analysis* 73 (2021), 102200.
[75] Sixiao Zhang, Hongxu Chen, Xiangguo Sun, Yicong Li, and Guandong Xu. 2022. Unsupervised graph poisoning attack via contrastive loss back-propagation. In *Proceedings of the ACM Web Conference 2022*. 1322–1330.
[76] Zeyu Zhu, Fanrong Li, Zitao Mo, Qinghao Hu, Gang Li, Zejian Liu, Xiaoyao Liang, and Jian Cheng. 2023. A^2Q: Aggregation-Aware Quantization for Graph Neural Networks. In *11th International Conference on Learning Representations*.
[77] Markus Zopf. 2022. 1-WL Expressiveness Is (Almost) All You Need. In *International Joint Conference on Neural Networks, IJCNN*. 1–8.
[78] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial Attacks on Neural Networks for Graph Data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2847–2856.
[79] Daniel Zügner and Stephan Günnemann. 2019. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *7th International Conference on Learning Representations*.

A PROOF OF PROPOSITION 3.3

Proof. We first prove by induction that the statement Equation (9) implies that there is a 1-to-1 correspondence between $\mathcal{J}_i$ and the isomorphism types of unfolding trees of height i, denoted by $\mathcal{T}_i$, for all $i \in \{0, \ldots, k\}$. In the base case $i = 0$, there is a single unfolding tree in $\mathcal{T}_0$ consisting of a single node. The uniform initialization $\mathcal{J}_0$ satisfies the requirement. Assume that φ is a bijection between $\mathcal{J}_i$ and $\mathcal{T}_i$, then the statement (9) together with the permutation-invariance guarantees that $f^{(i+1)}(A, \mathcal{A}) = f^{(i+1)}(B, \mathcal{B})$ if and only if $A = B$ and $\mathcal{A} = \mathcal{B}$. Hence, $\{\!\{\varphi(a) \mid a \in \mathcal{A}\}\!\} = \{\!\{\varphi(b) \mid b \in \mathcal{B}\}\!\}$, which uniquely determines an unfolding tree in $\mathcal{T}_{i+1}$ according to Definition 3.1. Vice versa, unfolding trees with different subtrees lead to distinguishable multisets. The result follows by Lemma 3.2 and the 1-to-1 correspondence shown above at layer k. □

B PROOF OF PROPOSITION 3.5

Proof. Assume $F_G^{(k)}(G_i) = F_G^{(k)}(G_j)$ but $C_l^{(k)}(G_i) \neq C_l^{(k)}(G_j)$. Then $f_G(\{\!\{F^{(k)}(v) \mid v \in V(G_i)\}\!\}) = f_G(\{\!\{F^{(k)}(v) \mid v \in V(G_j)\}\!\})$ but $\{\!\{c_l^{(k)}(v) \mid v \in V(G_i)\}\!\} \neq \{\!\{c_l^{(k)}(v) \mid v \in V(G_j)\}\!\}$. Because per construction $c_l^{(k)}(u) = c_l^{(k)}(v) \iff F^{(k)}(u) = F^{(k)}(v)$ it follows that $\mathcal{A} \neq \mathcal{B}$, contradicting Equation (11). □

C PROGRESSIVE BIT FLIP ATTACK

The PBFA on CNN weights is an attack that can crush a CNN by maliciously flipping minimal numbers of bits within its weight storage memory (i.e., DRAM). It was first introduced as an untargeted attack [48]. PBFA operates on integer quantized CNNs and seeks

Table 1: Preliminary case study illustrating the general vulnerability of GNNs to PBFA – pre- and post-attack mean of 10 runs of top-1 test accuracy (community) or AUROC (structure) of `INT8` quantized representative GNN architecture (GCN [63] with 3 layers, GAT [63] with 2 layers, GIN [68] with 5 layers) and dataset combinations (GCN on `Cora`, GAT on `CiteSeer`, GIN on `ogbg-mol`) baseline without BFAs; after PBFA [48] adapted to GNNs; after random bit flips (RBFA); total bit count of all model parameters (attack surface) in millions.

	COMMUNITY					STRUCTURAL				
Attack	**Dataset**	**Pre**	**Post**	**Flips**	**Bits**	**Dataset**	**Pre**	**Post**	**Flips**	**Bits**
RBFA	`Cora-GCN`	0.77	0.74	63	1.6M	`ogbg-molhiv-GIN`	0.71	0.53	953	15.1M
PBFA	`Cora-GCN`	0.77	0.12	9	1.6M	`ogbg-molhiv-GIN`	0.71	0.50	953	15.1M
RBFA	`CiteSeer-GAT`	0.58	0.48	63	3.8M	`ogbg-moltoxcast-GIN`	0.58	0.58	2662	16.6M
PBFA	`CiteSeer-GAT`	0.58	0.14	10	3.8M	`ogbg-moltoxcast-GIN`	0.58	0.57	2662	16.6M

to optimize Equation (15).

$$\max_{\{\widehat{\mathbf{W}}_q^{(l)}\}} \quad \mathcal{L}\Big(\Phi(\mathbf{X};\ \{\widehat{\mathbf{W}}_q^{(l)}\}_{l=1}^{L}),\ \mathbf{t}\Big) - \mathcal{L}\Big(\Phi(\mathbf{X};\ \{\mathbf{W}_q^{(l)}\}_{l=1}^{L}),\ \mathbf{t}\Big)$$
$$\text{s.t.} \quad \sum_{l=1}^{L} \mathcal{D}(\widehat{\mathbf{W}}_q^{(l)},\ \mathbf{W}_q^{(l)}) \ \in\ \{0, 1, \ldots, N_b\} \tag{15}$$

where $\mathbf{X}$ and $\mathbf{t}$ are input batch and target vector, $\mathcal{L}$ is a loss function, f is a neural network, L is the number of layers and $\widehat{\mathbf{W}}_q^{(l)}, \mathbf{W}_q^{(l)}$ are the perturbed and unperturbed integer quantized weights (stored in two's complement) of layer l. In the original work by [48], the function $\mathcal{L}$ used is the same loss originally used during network training. $\mathcal{D}(\widehat{\mathbf{W}}_q^{(l)}, \mathbf{W}_q^{(l)})$ represents the Hamming distance between clean- and perturbed-binary weight tensor, and N_b represents the maximum Hamming distance allowed through the CNN.

The attack is executed by flipping the bits along its gradient ascending direction w.r.t. the loss of CNN. That is, using the N_q-bits binary representation $\mathbf{b} = [b_{N_q-1}, \ldots, b_0]$ of weights $w \in \mathbf{W}_{q,l}$, first the gradients of $\mathbf{b}$ w.r.t. to inference loss $\mathcal{L}$ are computed

$$\nabla_{\mathbf{b}}\mathcal{L}\left[\frac{\partial\mathcal{L}}{\partial b_{N_q-1}}, \ldots, \frac{\partial\mathcal{L}}{\partial b_0}\right] \tag{16}$$

and then the perturbed bits are computed via $\mathbf{m} = \mathbf{b} \oplus (\text{sign}(\nabla_{\mathbf{b}}\mathcal{L})/2 + 0.5)$ and $\widehat{\mathbf{b}} = \mathbf{b} \oplus \mathbf{m}$, where $\oplus$ denotes the bitwise `xor` operator.

To improve efficiency over iterating through each bit of the entire CNN, the authors employ a method called progressive bit search (**PBS**). As noted earlier, we refer to this BFA variant employing PBS as Progressive BFA or PBFA. In PBS, at each iteration of the attack (to which we synonymously refer as *attack run*), in a first step for each layer $l \in [0, L]$, the n_b most vulnerable bits in $\widehat{\mathbf{W}}_q^{(l)}$ are identified through gradient ranking (in-layer search). That is, regarding input batch $\mathbf{X}$ and target vector $\mathbf{t}$, inference and backpropagation are performed successively to calculate the gradients of bits w.r.t. the inference loss and the bits are ranked by the absolute values of their gradients $\partial\mathcal{L}/\partial b$. In a second step, after the most vulnerable bit per layer is identified, the gradients are ranked across all layers s.t. the most vulnerable bit in the entire CNN is found (cross-layer search) and flipped. Should a PBS iteration not yield an attack solution, which can be the case if no single bit flip improves the optimization goal given in Equation (15), PBS is executed again and evaluates increasing combinations of 2 or more bit flips.

D A MOTIVATING CASE STUDY

Our preliminary case study summarized in Tab. 1 indicates a significant vulnerability of GNNs used in community-based tasks on graphs with strong homophily [51] to malicious BFAs such as PBFA, since it suggests a quantized GNN can be degraded so severely by an extremely small number of bit flips—relative to the network's attack surface—that it produces basically random output. In our case study, a GNN's output on the community-based tasks is random if its test accuracy drops below 14.3% (= 1/7) on `Cora`'s 7-class node classification task or below 16.7% (= 1/6) on `CiteSeer`'s 6-class node classification task. Our case study shows that this is consistently the case due to the PBFA adapted from [48] for all community-based architecture-dataset combinations examined. The number of bit flips required for completely degrading a GNN in a community-based task is remarkably small: tab. 1 shows that on average, the adapted PBFA flipped only 0.0004%of the total number of bits of the quantized GNNs' parameters. Regarding random bit flips (**RBFA**), the results of our case study are consistent with results obtained for full-precision GNNs [27] in that they demonstrate a relatively strong resilience of GNNs against such random perturbations.

On structural tasks on graphs with weak/low homophily as is typical in molecular, chemical, and protein networks [51] which are common in, e.g., drug development, PBFA is much less effective and degrades the network comparable to random bit flips. On the structural tasks in Table 1, a GNN's output is random if its test AUROC drops to 0.5. We found that on the `ogbg-moltoxcast` dataset, PBFA could not significantly degrade the network even after 2662 flips and that on `ogbg-molhiv`, 0,0063% of the total number of bits of the quantized GNNs' parameters had to be flipped by PBFA before the GNN's output was degraded to random output, which constitutes a 15.75 times increase compared to the community-based tasks. This increased resilience of GNNs trained on structural tasks compared to community-based tasks cannot be explained entirely by the higher number of GNN parameters found in the evaluated tasks requiring high structural expressivity, which mostly stems from the MLPs employed in GIN: The increase in required flips for PBFA to entirely degrade the network on the structural task is up to 2 orders of magnitudes larger compared to the community-based task, while the increase in the attack surface is at most 1 order of magnitude larger. Based on these observations, our work focuses on such structural tasks which are typically solved by GIN.

A.2 Crossfire: An Elastic Defense Framework

Title	Crossfire: An Elastic Defense Framework for Graph Neural Networks Under Bit Flip Attacks
Authors	Lorenz Kummer, Samir Moustafa, Wilfried Gansterer, Nils Kriege
Venue	Proceedings of the AAAI Conference on Artificial Intelligence (CORE ranking: A^*)
Identifier	DOI:10.1609/aaai.v39i17.33979
Division of work	Lorenz Kummer: Conceived the initial idea; formulated the defense method; implemented most components and designed experiments; conducted the evaluation; wrote most of the article and created the visualizations; produced the conference poster; reviewed the related work. Samir Moustafa: Implemented the underlying quantization package; participated in regular quantization discussions and occasional pair-programming sessions; wrote the quantization-related sections; reviewed and refined the article, conference poster. Wilfried Gansterer: Participated in discussions; reviewed and refined the article and conference poster; provided mentoring and supervision. Nils Kriege: Participated in discussions; reviewed and refined the article and conference poster; provided mentoring and supervision.

The Thirty-Ninth AAAI Conference on Artificial Intelligence (AAAI-25)

Crossfire: An Elastic Defense Framework for Graph Neural Networks Under Bit Flip Attacks

Lorenz Kummer[1,2], **Samir Moustafa**[1,2], **Wilfried Gansterer**[1], **Nils Kriege**[1,3]

[1]Faculty of Computer Science, University of Vienna, Vienna, Austria
[2]Doctoral School Computer Science, University of Vienna, Vienna, Austria
[3]Research Network Data Science University of Vienna, Vienna, Austria
{lorenz.kummer, samir.moustafa, wilfried.gansterer, nils.kriege}@univie.ac.at

Abstract

Bit Flip Attacks (BFAs) are a well-established class of adversarial attacks, originally developed for Convolutional Neural Networks within the computer vision domain. Most recently, these attacks have been extended to target Graph Neural Networks (GNNs), revealing significant vulnerabilities. This new development naturally raises questions about the best strategies to defend GNNs against BFAs, a challenge for which no solutions currently exist. Given the applications of GNNs in critical fields, any defense mechanism must not only maintain network performance, but also verifiably restore the network to its pre-attack state. Verifiably restoring the network to its pre-attack state also eliminates the need for costly evaluations on test data to ensure network quality. We offer first insights into the effectiveness of existing honeypot- and hashing-based defenses against BFAs adapted from the computer vision domain to GNNs, and characterize the shortcomings of these approaches. To overcome their limitations, we propose Crossfire, a hybrid approach that exploits weight sparsity and combines hashing and honeypots with bit-level correction of out-of-distribution weight elements to restore network integrity. Crossfire is retraining-free and does not require labeled data. Averaged over 2,160 experiments on six benchmark datasets, Crossfire offers a 21.8% higher probability than its competitors of reconstructing a GNN attacked by a BFA to its pre-attack state. These experiments cover up to 55 bit flips from various attacks. Moreover, it improves post-repair prediction quality by 10.85%. Computational and storage overheads are negligible compared to the inherent complexity of even the simplest GNNs.

Introduction

Graph Neural Networks (GNNs) are effective machine learning methods for processing structured data in graph format, consisting of nodes and edges. They demonstrate versatility by enabling the application of deep learning in diverse domains such as finance, social networks, medicine, chemistry, and biological data analysis (Lu and Uddin 2021; Cheung and Moura 2020; Sun et al. 2021; Wu et al. 2018; Xiong et al. 2021). As GNNs become more widely used, it is crucial to examine their potential security vulnerabilities. Conventional adversarial attacks on GNNs primarily involve manipulating input graph data (Wu et al. 2022). These attacks include poisoning, which leads to the learning of flawed models (Ma, Ding, and Mei 2020; Wu et al. 2022), and evasion strategies, which use adversarial examples to impair inference. Such attacks on GNNs, which involve altering node features, edges, or introducing new nodes (Sun et al. 2020; Wu et al. 2022), as discussed in prior studies (Ma, Ding, and Mei 2020), can be either targeted or untargeted. Targeted attacks reduce the model's prediction quality on specific instances, whereas untargeted attacks affect the model's overall performance (Zhang et al. 2022a). For a thorough overview of graph poisoning and evasion attacks, along with defenses and relevant algorithms, refer to the detailed reviews by Jin et al. (2021) and Dai et al. (2022).

Recent research on GNN vulnerability has expanded beyond poisoning and evasion attacks on input graphs, now encompassing systematic attacks that directly manipulate network weights through malicious bit flips during inference. While such attacks are well-understood for Convolutional Neural Networks (CNNs) in computer vision (Rakin, He, and Fan 2019; Qian et al. 2023; Khare et al. 2022), understanding GNN vulnerability to BFAs is a relatively new area of study. Currently, only one dedicated BFA for GNNs has been described by Kummer et al. (2024), and existing defenses against BFAs (Khare et al. 2022) have not been evaluated for GNNs. To the best of our knowledge, the topic of defending GNNs against BFAs has not yet been addressed in the literature.

Given the fundamental differences between GNNs and CNNs – such as GNNs' reliance on the message-passing algorithm (MP) to process graph-structured data and the lack of vulnerable convolutional filters (Hector et al. 2022) – as well as distinct properties like expressivity that can be exploited by attackers (Kummer et al. 2024), it is essential to explore whether existing CNN defenses against BFAs are applicable to GNNs. Moreover, it is crucial to develop defenses specifically designed for GNNs, given the importance of their applications across multiple domains.

In the following, we assume a white-box threat model to evaluate CNN defenses against BFAs on GNNs and introduce our approach, considering an attacker who can precisely manipulate bits without budget constraints.

Related Work The BFA initially presented in Rakin, He, and Fan (2019), targets a quantized CNN by conducting

a single forward-backward pass using a randomly selected training data batch, without updating the weights. It identifies the top-k binary gradient bits as potential bit-flip candidates. These bits are then flipped iteratively to maximize the loss (using the same loss function as in training) until the desired network degradation is achieved. This original BFA is termed Progressive BFA (PBFA). We use BFA to refer to a broader range of attacks that systematically induce malicious bit flips in a neural network's weights and biases, and refer the interested reader to the comprehensive survey for BFAs on CNNs by Qian et al. (2023). For GNNs, only a single dedicated BFA has been explored so far, termed Injectivity Bit Flip Attack (IBFA), which exploits certain properties related to GNN expressivity by Kummer et al. (2024).

To defend against BFAs on CNNs, several approaches have been proposed. Network hardening methods, such as adversarial (He et al. 2020; Li et al. 2020a) and perturbation-resilient training (Chitsaz et al. 2023), as well as gradient obfuscation strategies (Zhang et al. 2022b; Wang et al. 2023a) , focus on resisting gradient-based bit search attacks. However, they may require (re)training, which limits their ex-post deployment, and they do not necessarily rectify compromised neurons. Detection-focused methods leverage hashing (Javaheripi and Koushanfar 2021; Li et al. 2021) or output code matching (Özdenizci and Legenstein 2022) for efficient bit flip detection and consistently achieve high detection rates in the computer vision domain. These methods often wrap around existing pre-trained networks and can be applied ex-post. However, their restoration capabilities are limited, often necessitating retraining or pruning of manipulated weights. Proactive defenses take a different approach by anticipating attacks through analyzing assumed underlying attack mechanisms (Liu et al. 2023; Li et al. 2020a). These methods prioritize weight restoration over pruning, aiming to revert the network to its pre-attack configuration (Liu et al. 2023), or use statistical approximation to address compromised weights (Li et al. 2020a). For an overview of defense mechanisms, see the review by Khare et al. (2022).

Challenges and Limitations Existing defenses against BFAs face shared challenges and limitations. First, they lack mechanisms to ensure full network restoration after attacks; while prediction quality may recover, network integrity is not guaranteed. Reloading a clean model or delegating control to a backup system after attack detection is often inefficient and delay-prone (Li et al. 2021). Second, reliance on a single defense mechanism leaves these approaches susceptible to loophole-exploiting BFAs (Liu et al. 2023).

Moreover, the transferability of defense strategies from CNNs to GNNs remains unexplored. Given GNNs' critical roles in applications like medical diagnosis (Li et al. 2020b; Lu and Uddin 2021), health record modeling (Liu et al. 2020; Sun et al. 2021), and drug development (Xiong et al. 2021; Cheung and Moura 2020), ensuring their authenticity post-attack is essential. A robust defense strategy must integrate multiple protective layers and incorporate efficient, low-cost verification.

Contribution

(1) We study two state-of-the-art BFA defense methods based on honeypots (Liu et al. 2023) and hashing (Li et al. 2021), which were initially developed for CNNs, and assess their effectiveness for GNNs, identifying notable weaknesses in their application to this domain. **(2)** We propose an innovative honeypot selection method using unlabeled data to enhance detection rates and validate GNN reconstruction using a robust, well-known and studied, yet lightweight hash function. **(3)** Moreover, we build upon existing work and introduce an efficient weight group-based checksum approach to detect hits in non-honeypot weights, addressing the limitation of solely relying on honeypots. This approach, together with the bit-level correction of out-of-distribution (OOD) weight elements as well as the careful exploitation of the relationship between BFAs and GNN weight sparsity, allows us to reconstruct the network in certain cases even when hits on the preselected honeypots are avoided by a diligent attacker.

Our work establishes a strong and dependable defense framework, obviating the need for post-reconstruction quality verification of the GNN.

Preliminaries

This section provides an overview and detailed description of the key elements essential to our research: GNNs, BFAs, the threat scenario, as well as the honeypot-based NeuroPots (Liu et al. 2023) and the hashing-based runtime Adversarial Weight Attack Detection and Accuracy Recovery (RADAR) defense methods (Li et al. 2021).

Graph Neural Networks GNNs leverage graph structure and node features to derive representation vectors for specific nodes, denoted as $\mathbf{h}_v$ for node v, or for the entire graph G, denoted as $\mathbf{h}_G$. Contemporary GNNs use a neighborhood aggregation or message-passing approach, where the representation of a node is iteratively updated by aggregating the representations of its neighboring nodes. Upon completion of k layers of aggregation, the representation of a node encapsulates the structural information within its k-hop neighborhood (Xu et al. 2019; Welling and Kipf 2016). The kth layer of a GNN computes the node features $\mathbf{h}_v^k$ as defined by

$$\mathbf{a}_v^k = \textsc{Aggregate}^k\left(\{\!\{\mathbf{h}_u^{k-1} \mid u \in N(v)\}\!\}\right)$$
$$\mathbf{h}_v^k = \textsc{Combine}^k\left(\mathbf{h}_v^{k-1}, \mathbf{a}_v^k\right)$$

with node neighborhood $N(v)$ and multiset $\{\!\{\}\!\}$. Initially, $\mathbf{h}_v^0$ represents the features of node v in the given graph. As the choice of $\textsc{Aggregate}^k$ and $\textsc{Combine}^k$ in GNNs is critical, several variants have been proposed (Xu et al. 2019).

Quantization Quantization reduces the precision or adopts efficient representations for weights, biases, and activations, decreasing model size and memory usage (Jacob et al. 2018; Kummer et al. 2023).

In accordance with the typical setup chosen in related work on BFA, we apply scale quantization to map `FLOAT32` tensors to the `INT8` range. Such a quantization function $\mathcal{Q}$

and its associated dequantization function $\mathcal{Q}^{-1}$ are

$$\begin{aligned} \mathcal{Q}(\mathbf{W}^l) &= \mathbf{W}_q^l = \text{clip}(\lfloor \mathbf{W}^l/s \rceil,\ a, b), \\ \mathcal{Q}^{-1}(\mathbf{W}_q^l) &= \widehat{\mathbf{W}}^l = \mathbf{W}_q^l \times s. \end{aligned} \tag{1}$$

Here, s denotes the scaling parameter, $\text{clip}(x, a, b) = \min(\max(x, a), b)$ with a and b the minimum and maximum thresholds (also known as the quantization range), $\lfloor \dots \rceil$ denotes nearest integer rounding, $\mathbf{W}^l$ is the weight of a layer l to be quantized, $\mathbf{W}_q^l$ its quantized counterpart and $\widehat{\dots}$ indicates a perturbation, i.e., rounding errors in the case of (1). Similar to other works on BFA that require quantized target networks such as (Rakin, He, and Fan 2019), we address this issue of non-differentiable rounding and clipping functions present in $\mathcal{Q}$ by using Straight Through Estimation (STE) (Bengio, Léonard, and Courville 2013).

Bit Flip Attacks PBFA, introduced in the seminal work by Rakin, He, and Fan (2019), uses a quantized trained CNN Φ and progressive bit search (PBS) to identify bits for flipping. PBS starts with a forward and backward pass, performing error backpropagation without updating weights on a randomly selected batch $\mathbf{X}$ of training data with a target vector $\mathbf{t}$. It then selects the weights corresponding to the top-k largest binary encoded gradients as potential candidates for bit flipping. These candidate bits are iteratively tested across all L layers to find the bit that maximizes the difference between the loss $\mathcal{L}$ of the perturbed CNN and the loss of the unperturbed CNN,

$$\max_{\{\widehat{\mathbf{W}}_q^l\}} \mathcal{L}\Big(\Phi(\mathbf{X};\ \{\widehat{\mathbf{W}}_q^l\}_{l=1}^L),\ \mathbf{t}\Big) - \mathcal{L}\Big(\Phi(\mathbf{X};\ \{\mathbf{W}_q^l\}_{l=1}^L),\ \mathbf{t}\Big)$$

whereby the same $\mathcal{L}$ is used that was minimized during training, e.g., (binary) cross entropy (B)CE for (binary) classification). Using ℓ_1 or Kullback-Leibler-Divergence (Kullback and Leibler 1951) for $\mathcal{L}$, the variant IBFA by Kummer et al. (2024) dedicated to GNNs instead optimizes

$$\min_{\{\widehat{\mathbf{W}}_q^l\}} \mathcal{L}\Big(\Phi(\mathbf{X}_a; \{\widehat{\mathbf{W}}_q^l\}_{l=1}^L),\ \Phi(\mathbf{X}_b; \{\widehat{\mathbf{W}}_q^l\}_{l=1}^L)\Big)$$

via PBS and uses a unique input data selection process related to its theoretical fundament given by

$$\arg\max_{\{\mathbf{X}_a, \mathbf{X}_b\}} \mathcal{L}\Big(\Phi(\mathbf{X}_a; \{\mathbf{W}_q^l\}_{l=1}^L),\ \Phi(\mathbf{X}_b; \{\mathbf{W}_q^l\}_{l=1}^L)\Big).$$

IBFA targets specific mathematical properties of GNNs that are crucial for graph learning tasks requiring high structural expressivity. Kummer et al. (2024) demonstrate IBFA's efficacy in exploiting GNN expressivity, rendering GNNs indifferent to graph structures and compromising their predictive quality in tasks that require structural discrimination.

NeuroPots Liu et al. (2023) introduce NeuroPots, a proactive defense mechanism for CNNs that embeds honey neurons as vulnerabilities to lure attackers and facilitate fault detection and model restoration. The proportion of honey neurons per layer is controlled by the hyperparameter p, indicating the percentage of neurons modified. The framework uses a checksum-based detection method, anticipating that most bit flips will target these trapdoors, and uses trapdoor refreshing to recover the model's prediction quality.

NeuroPots provides two methods: retraining-based and heuristic. The retraining approach, suitable for those with ample training data, is data-intensive. The heuristic method is more efficient, enhancing specific neuron activation for trapdoor embedding and adjusting adjacent weights to maintain its contribution to the next layer. This method is ideal for full-precision models, though it may cause minor quantization errors. It can be applied to any layer, including direct modifications of honey neuron activations in the input layer (Liu et al. 2023). The one-shot encoding process used by NeuroPots can be described as follows:

$$o_i^{l+1} = \sum_{j=1}^{n^l} w_{ji}^l \cdot o_j^l = w_{0i}^l \cdot o_0^l + \dots + \left(\frac{1}{\gamma} \cdot w_{hi}^l\right)(\gamma \cdot o_h^l)$$

where o_h^l denotes the honey neuron at layer l, w_{hi}^l denotes the associated honey weights, and γ the rescaling factor. For a typical neuron, its influence on the output of the next layer in the presence of BFAs can be formulated as $o^{l+1} = (w + \Delta w) \cdot o^l$, where Δw denotes the weight distortion arising from bit flips. Conversely, considering the one-shot trapdoor as an example, the impact of a honey neuron on the subsequent layer can be represented as:

$$o^{l+1} = \left(\frac{1}{\gamma} \cdot w + \Delta w\right) \cdot \gamma \cdot o^l = (w + \Delta w) \cdot o^l + (\gamma - 1) \cdot \Delta w \cdot o^l$$

Obviously, o^{l+1} experiences an increase of $(\gamma - 1) \cdot \Delta w \cdot o^l$ in comparison to a regular neuron. Furthermore, attackers tend to flip the most significant bits of weights, leading to a substantial perturbation Δw. Consequently, the impact of the honey neuron on o^{l+1} becomes more pronounced, particularly for larger values of γ. This alteration propagates and accumulates across subsequent layers, causing a substantial shift in the model's output.

NeuroPots, despite its novelty, has limitations. Random honeypot selection may overlook key neurons or include inactive ones, while a global scaling parameter γ ignores neuron-specific roles. Critically, it can not certify restoration to the pre-attack state, requiring post-recovery evaluation, as it fails to detect changes in non-honeypot neurons.

RADAR RADAR (Li et al. 2021) is designed to protect CNN weights from PBFA. It organizes weights into groups within each layer and uses a checksum-based algorithm to generate a 2-bit to 3-bit signature per group. During runtime, this signature is computed and compared with a stored reference to detect BFAs. If detected, the weights in the affected group are zeroed to minimize prediction quality loss. RADAR is integrated into the inference computation stage.

RADAR efficiently detects most bit flips, including random ones, but only partially restores prediction quality by zeroing attacked weight groups. It cannot fully revert the network to its pre-attack state, requiring test data evaluation to confirm recovery. For large numbers of bit flips, zeroing further degrades the network and impacts non-attacked weights, causing collateral damage.

Threat Scenario In line with the general trend in literature on BFAs for CNNs, e.g., (Yao, Rakin, and Fan 2020; Rakin, He, and Fan 2019; He et al. 2020; Li et al. 2020a, 2021; Liu et al. 2023; Javaheripi and Koushanfar 2021) as well as IBFA (Kummer et al. 2024), we assume that the target network is `INT8` quantized. Furthermore, we adopt the assumption which is common in related work that the attacker has the capability to exactly flip the bits chosen by the bit-search algorithm through mechanisms such as RowHammer (Mutlu and Kim 2019), NetHammer (Lipp et al. 2020) or others (Breier et al. 2018). For a detailed discussion on the technical aspects of implementing arbitrary flips of the identified vulnerable bits in hardware, we refer to Wang et al. (2023a).

To test the reliability of our framework under worst-case conditions, we assume that the attacker is not subjected to budget considerations (Hector et al. 2022), although typically, flipping more than 25 bits is considered difficult for an attacker (Yao, Rakin, and Fan 2020) and 50 bits is considered an upper boundary (Wang et al. 2023a). Moreover, we assume that some amount of training data as well as information on the network structure is available to the attacker, which is a typical assumption in related work on BFAs, e.g., Liu et al. (2023). This information can be acquired through methods such as side-channel attacks (Yan, Fletcher, and Torrellas 2020; Batina et al. 2018). Together, these typical assumptions amount to a white box threat model, whereby the attacker's goal is to crush a well-trained and deployed quantized GNN via BFA.

In our defense strategy, we presume the presence of unlabeled data and secure storage of original honey weights and hashes in an inaccessible location for potential attackers. Similar to (Liu et al. 2023), we argue that such a secure storage can be realized via trusted execution environments (TEE) like Intel SGX (McKeen et al. 2016) or ARM Trustzone (Pinto and Santos 2019; ARM 2009).

The Crossfire Defense Mechanism

Crossfire is a rapid detection and recovery method that restores model prediction quality and verifies if the recovered model matches the pre-attack state, eliminating the need for test data. It operates in three stages: initialization, monitoring, and reconstruction. In initialization, Crossfire processes a fully trained, quantized GNN, inducing sparsity through dequantization and ℓ_1 pruning. Honeypots and scaling parameters are computed from gradients and network depth, followed by re-quantization and generation of hashes for layers, rows, and columns. During monitoring, layer hashes detect alterations, triggering the reconstruction phase. In reconstruction, altered weights are identified via row and column hashes and are either restored with honeypots, corrected at the bit level (if OOD), or zeroed if the other options fail. The induced sparsity increases the likelihood that zeroing resets the GNN to its pre-attack state, with layer hashes finally verifying the GNN's integrity.

Inducing sparsity We exploit that BFAs are prone to flip most significant bits of near-zero weights (Qian et al. 2023; Li et al. 2021) by inducing sparsity via ℓ_1-magnitude pruning, setting $w \in \mathbf{W}^l$ to zero if $|w| < \tau$, where τ is the p-th quantile $\Lambda_p(|\mathbf{W}^l|)$, pruning $p{\cdot}100\%$ of the smallest weights:

$$w \leftarrow \begin{cases} 0 & \text{if } |w| < \tau, \\ w & \text{otherwise.} \end{cases}$$

This increase in the attack surface will steer attackers toward neurons with high sparsity and help mask the most important neurons by increasing the number of potential target weights, saturating the network with zero weights. Theoretical insights from (Frankle and Carbin 2018; Malach et al. 2020) suggest that introducing a limited amount of sparsity to a neural network does not significantly reduce its prediction quality. Further, by guiding attackers towards ℓ_1-pruned weights, subsequent checksum-based zeroing will allow reconstruction to the pre-attack state. While a sophisticated attacker might avoid targeting zeroed weights, this strategy reduces the attack's efficacy by narrowing viable options as sparsity increases, forcing less optimal bit flips.

Honeypot Selection Focusing on the heuristic approach due to the need for labeled data in Liu et al. (2023)'s retraining method, we observe some critical aspects. Randomly selecting neuron honeypots, as proposed to counter activation-ranking attackers, ignores slight variations in neuron activation across batches. Effective deployment would thus require dynamic γ and honeypot adjustment or a batch-averaging strategy. However, dynamic adjustment risks exposing honeypots via network changes, while batch averaging distributes honeypots across rankings, enhancing effectiveness even against advanced attackers.

Thus to improve honeypot selection, we employ a two-fold method: First, we create predictions for a small subset of unlabeled data S. We derive classes from these predictions to form a set P of tuples $(\mathbf{t}, \mathbf{X})$, where $\mathbf{X}$ is the input data (batch) and $\mathbf{t}$ are the predicted classes ($\arg\max$ of class probabilities), serving as pseudo labels for backpropagation.

$$P = \Big\{ (\mathbf{t}, \mathbf{X}) \mid \mathbf{t} = \underset{axis=1}{\arg\max} \left(\Phi(\mathbf{X};\ \{\mathbf{W}^l\}_{l=1}^{L}), \mathbf{X} \in S\right) \Big\}$$

For each $(\mathbf{t}, \mathbf{X}) \in P$, we conduct one pass of backpropagation through the GNN to obtain gradients for the weights $\mathbf{W}^l$ using an appropriate loss function, e.g, (B)CE loss for (binary) classification. For each $\mathbf{W}^l$, the gradients obtained for each data batch are accumulated and represented as $\mathbf{G}^l$, which is defined as

$$\mathbf{G}^l = \sum_{(\mathbf{t},\mathbf{X}) \in P} \frac{\partial \mathcal{L}\Big(\Phi(\mathbf{X};\ \{\mathbf{W}^l\}_{l=1}^{L})\Big), \mathbf{t}\Big)}{\partial \mathbf{W}^l}.$$

For each layer, we select the set of honeypot neurons N_k^l from all neurons N by choosing the indices of the top-k largest sums of accumulated absolute neuron gradients, where $k = n^l \cdot p$. Here, n^l is the number of neurons in the layer, p is the base percentage of honeypot neurons, and $axis = 1$ indicates per-neuron summation. Note, that access to unlabeled data is a conservative assumption. Labeled data could enhance the defense by improving gradient-based identification of vulnerable weights. Notably, BFAs

also rarely use labeled data; IBFA, the only GNN-specific BFA, avoids labeled data entirely (Kummer et al. 2024).

$$N_k^l = \left\{ \arg\operatorname{top} k \left(\sum_{axis=1} |\mathbf{G}^l| \right) \right\}$$

Choice of Scaling Parameters Liu et al. (2023) highlight that neuron rescaling can introduce quantization errors, with each neuron's unique weight distribution requiring tailored rescaling to minimize these errors. Such quantization-induced perturbations propagate through matrix multiplications , with their impact on network performance depending on each neuron's contribution to classification (Xiaoqin Zeng and Yeung 2001; Xiang et al. 2019). A uniform scaling parameter for all honeypots, as suggested by Liu et al. (2023), is thus suboptimal.

We propose individualized scaling for each neuron to better minimize quantization error and preserve classification performance. Initially, we augment the scaling factor γ proportionally to the network's depth using a scaling factor $\lambda \geq 1$ to obtain a layer's scaling $\gamma^l = \gamma \cdot \lambda^l$. This approach offers two benefits. First, it generally reduces propagated error as deeper layers are scaled higher. In GNNs, this favors distinguishing between nodes based on local neighborhoods, as it directs attackers towards deeper layers. Given that the evaluated GNNs may be shallower than the graph diameter, this strategy is consistent with using shallow architectures to prevent overfitting (Morris, Fey, and Kriege 2021) and over-smoothing (Liu, Gao, and Ji 2020).

Further, we do not use γ^l directly to rescale the honey neurons. Instead, we compute a saliency vector $\bar{\mathbf{s}}^l$ for each layer that considers the accumulated gradients magnitude

$$\bar{\mathbf{s}}^l = 1 + \frac{(\mathbf{s}^l - \min(\mathbf{s}^l)) \cdot (\gamma^l - 1)}{\max(\mathbf{s}^l) - \min(\mathbf{s}^l)}, \mathbf{s}^l = \sum_{axis=1} |\mathbf{G}^l_{N_k^l}|.$$

The saliency vector $\mathbf{s}^l$ is then multiplied with the honeypots N_k^l using per-column division, $\mathbf{W}^l_{q,N_k^l} \oslash_1 \bar{\mathbf{s}}^l$ and the inputs are scaled using per-row product $\mathbf{X}^l_{N_k^l} \odot_0 \bar{\mathbf{s}}_l$.

Out of Distribution Weights Quantization can lead to limited range, i.e. for `INT8` (stored in two's complement) $\exists L^l, U^l, -128 \leq L^l \leq \min(\mathbf{W}_q^l) \wedge \max(\mathbf{W}_q^l) \leq U^l \leq 127$. BFAs typically target zero or near-zero weights and flip the most significant bit (MSB) of such weights (Li et al. 2020a), which has a high likelihood of creating an OOD element within that particular weight tensor. That is, if we choose $L^l = \min(\mathbf{W}_q^l)$ and $U^l = \min(\mathbf{W}_q^l)$, it holds that $\forall \widehat{w}_{ij} \in \widehat{\mathbf{W}}_q^l, w_{ij} \in \mathbf{W}_q^l | L^l > \widehat{w}_{ij} \vee U^l < \widehat{w}_{ij} \Rightarrow \widehat{w}_{ij} \neq w_{ij}$, indicating that $\widehat{w}_{ij}$ was subjected to a bit flip pushing it out of $\mathbf{W}_q^l$ range. Under the assumption that double flips in a single $\widehat{w}_{ij}$ are uncommon and that an attacker flips the (or one of the) MSBs, potentially iteratively unsetting MSB-n bits in $\widehat{w}_{ij}$ until $L^l \leq \widehat{w}_{ij} \leq U^l$ will either entirely undo the attacker's bit flips or at least reduce their effect. For example, if we observe $L^l = -50$ (`11001110`), $U^l = 60$ (`00111100`) and $\widehat{w}_{ij} = -58$ (`11000110`), iteratively unsetting $\widehat{w}_{ij}$'s two MSB's will lead to 70 (`01000110`) and 6 (`00000110`), which, under the assumption an attacker preferably flips MSBs in near zero weights, could be a correct reconstruction for w_{ij}. Naturally, this approach has its own limitations, as, e.g. $\widehat{w}_{ij} = 14$ (`00001110`) would neither have been detected nor corrected for above U^l, L^l. However, as it is only complementary to our honeypots and checksum-based detection approach, it allows for a degree of reconstruction that would not be possible by simply replacing non-honeypot weights by zeros (Li et al. 2021) or statistical approximations of w_{ij} (Li et al. 2020a).

Attack Detection and Reconstruction For attack detection and network verification, we opted for `Blake2b` (Aumasson et al. 2013), a widely recognized hash function with established security analyses (Guo et al. 2014; Rao and Prema 2019) and documented resource constraints (Sugier 2017). `Blake2b` excels in speed and security, outperforming SHA-3. Its parallel compression and adaptable parameters make it versatile for various applications, confirming its importance in contemporary cryptographic practices. Specifically, we compute the d-byte hashes for each row and column of a matrix $\mathbf{W}_q^l$ of size $n \times m$ with $d = 2$

$$\text{ColumnHash}(j, d) = \texttt{Blake2b}\left(\sum_{i=1}^{m} \mathbf{W}^l_{q,ij} \right)$$

$$\text{RowHash}(i, d) = \texttt{Blake2b}\left(\sum_{j=1}^{n} \mathbf{W}^l_{q,ij} \right)$$

and store them in vectors $\mathbf{rh}^l, \mathbf{ch}^l$. The storage overhead for this is negligible relative to the storage costs of the network. If desired, d (to which we, in this context, refer to as *cross digest*) can also be chosen dynamically by, e.g., matrix size, $d = \min\left(\max\left(1, \frac{\log_2(n \cdot m)}{8}\right), M\right)$ with M denoting the desired maximum number of bytes. Our hashing approach allows us to pinpoint changes to single elements in the matrix and surgically correct or remove them. This provides an advantage over RADAR, which can only detect changes in an entire group of weights and subsequently zeros them out, and it surpasses NeuroPots by enabling the detection of changes in non-honeypot weights. If we compute $\mathbf{rh}^l, \mathbf{ch}^l$ for $\mathbf{W}_q^l$ and, at some point after an attack on $\mathbf{W}_q^l$, obtain $\widehat{\mathbf{rh}}^l, \widehat{\mathbf{ch}}^l$ for $\widehat{\mathbf{W}}_q^l$, then $\widehat{\mathcal{R}}^l = \{i \mid i \in [1, n], \mathbf{rh}_i^l \neq \widehat{\mathbf{rh}}_i^l\}$ and $\widehat{\mathcal{C}}^l = \{i \mid i \in [1, m], \mathbf{ch}_i^l \neq \widehat{\mathbf{ch}}_i^l\}$ will be the row and column indices of perturbed elements in $\widehat{\mathbf{W}}_q^l$. For the elements located at these indices, repair mechanisms are executed. They are either replaced with honeypot weights (if they were a honeypot), have their distribution repaired by unsetting MSBs (if they were OOD), or are zeroed if none of the other options leads to positive layer verification. Note the initial sparsity induction via pruning increases the likelihood that zeroing resets weights to their pre-attack state.

Post Reconstruction Verification All weight matrices are hashed using the `Blake2b` hash function with a digest size of 4 bytes (referred to as *layer digest*). This setup detects changes in the weights matrix with high reliability, as

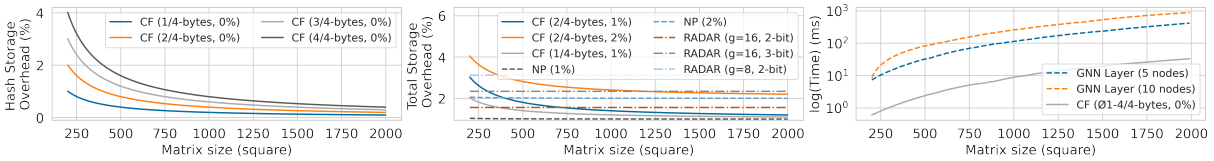


Figure 1: Storage overhead of Crossfire (CF) hashes relative to matrix size (INT8), varying cross digest sizes (left). Storage overhead of CF relative to matrix size, varying cross digest sizes and honeypot percentages (%) compared to RADAR and NeuroPots (NP) (center). Average hashing times (milliseconds) for CF across different cross digest sizes, plotted against the time complexity of a simple INT8 GNN layer (right) for 5 and 10 node graphs. Layer digest sizes fixed at 4 bytes.

Name	Graphs	Nodes	Edges	Tasks	Metric
pcba	437.9k	26.0	28.1	128	AP
muv	93.1k	24.2	26.3	17	AP
hiv	41.1k	25.5	27.5	1	AUROC
tox-cast	8.6k	18.5	19.3	617	AUROC
tox21	7.8k	18.6	19.3	12	AUROC
bace	1.5k	34.1	36.9	1	AUROC

Table 1: Overview of the six OGB (Hu et al. 2020) benchmark ogbg-mol datasets used: number of graphs and tasks, average number of nodes, edges and recommended metric.

we demonstrate experimentally below. Although Blake2b with a 4-byte digest is more computationally expensive than simpler hashes used by, e.g., RADAR, it is only executed to detect an attack, triggering Crossfire's simpler, more granular mechanisms, and to verify post-attack reconstruction.

Experiments

Our experiments[1] are designed to test the following hypotheses regarding Crossfire's performance on GNNs: **(a)** Crossfire has a higher chance of detecting bit flips compared to other methods, **(b)** Crossfire can repair the GNN so that it is indistinguishable from its pre-attack state, **(c)** Crossfire results in a smaller difference between pre-attack and post-reconstruction GNN performance, and **(d)** Crossfire does not increase the vulnerability of GNNs to evaluated BFAs.

We evaluate Crossfire against basic PBFA as well as IBFA, an attack specifically designed for GNNs with no known defense. In the context of our experiments, an attack is considered detected if a single bit flip is identified by the detection mechanism of the respective defense strategy (i.e., RADAR, NeuroPots, or Crossfire). A network is considered reconstructed if applying the Blake2b hash function with a 4-byte digest to all weight matrices indicates they are identical to their pre-attack state. We optimize Crossfire's and NeuroPots' hyperparameters via grid search for the number of honeypots $p \in \{0.01, 0.05, 0.1\}$ (given as percentage of the number of neurons) and the (initial) rescaling factor $\gamma \in \{1.33, 1.66, 2.0\}$. We fix Crossfire's depth rescaling parameter at $\lambda = 1.1$ and ℓ_1 pruning ratio at 75% and compute gradients from 10 random samples from each dataset's training split. Note that, while the assumption of having unlabeled data falling into the same distribution as the original training data may not always hold in practice, selecting samples approximating the original training distribution might be a feasible substitute. For RADAR, we used group size 16 and 2-bit hashes, as recommended by its authors.

[1] https://github.com/lorenz0890/crossfireaaai2025

Datasets and Models The same six Open Graph Benchmark (OGB) graph classification benchmark datasets are chosen for evaluation as in Kummer et al. (2024)'s work on IBFA. The goal in each dataset is to predict properties based on molecular graph structures, such that the datasets are consistent with the underlying assumptions of IBFA described by Kummer et al. (2024). The datasets are split using a scaffold-based strategy, to ensure structural diversity between subsets (Wu et al. 2018; Hu et al. 2020). Dataset characteristics are summarized in Table 1. Area under the receiver operating characteristic curve (AUROC) or average precision (AP) are used to measure prediction quality, as recommended by Hu et al. (2020).

To remain consistent with Kummer et al. (2024), we use 5-layer Graph Isomorphism Networks (GIN) quantized to INT8 via scale quantization. GIN is widely adopted, integrated into popular frameworks (Fey and Lenssen 2019), and applied in research (Wang et al. 2023b; Gao et al. 2022).

Results

We conducted an extensive evaluation of Crossfire, consisting of 2,160 individual experiments across six datasets, using a system with a NVIDIA GTX 1650 GPU (4GB VRAM) and an Intel(R) i7-10750H CPU (64GB RAM).

Attack Detection and Mitigation For basic PBFA, Crossfire, NeuroPots, and RADAR achieved nearly 100% attack detection across all datasets. Crossfire demonstrated superior detection of individual bit flips, outperforming competitors in most cases (Figure 2). Notably, Crossfire repaired damage to the GNN in over 50% of the cases for up to 25 bit flips, significantly outperforming NeuroPots. By design, RADAR cannot reverse bit-flip-induced damage to restore a GNN to its pre-attack state. While selecting honeypots based on gradients could theoretically guide BFAs toward vulnerable areas and potentially lower post-attack GNN prediction quality, this effect was not significant (see below). Crossfire compensates with exceptional restoration performance, reliably restoring prediction quality to 100% for up to 25 bit flips and outperforming the best competing method, which

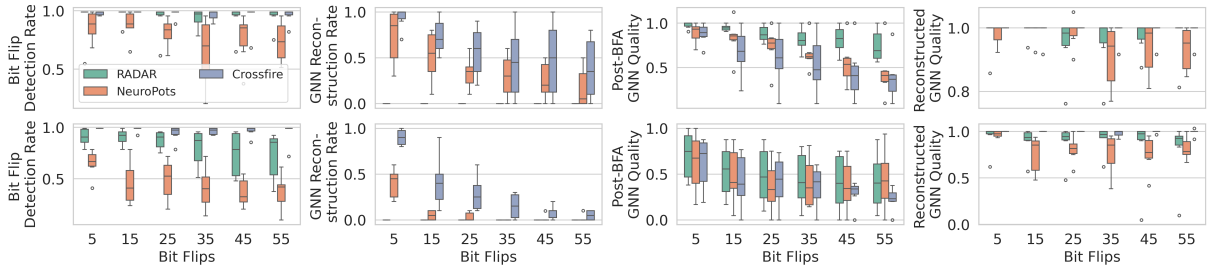


Figure 2: Post-BFA and reconstructed GNN prediction qualities (with pre-attack normalized to 100% to account for different scale quality metrics AP and AUROC), reconstruction, and detection rates for GNNs under PBFA (top row) and IBFA (bottom row) with varying amounts of bit flips, defended by NeuroPots, RADAR and Crossfire. In each subplot, a box represents data aggregated from 6 datasets, with 10 runs per dataset, totaling 1080 runs for each IBFA and PBFA at optimal hyperparameters.

reached 98.6%. Crossfire also restored prediction quality up to 98.1% for up to 55 PBFA-induced bit flips.

Regarding IBFA (Figure 2), Crossfire's performance is slightly degraded compared to the PBFA attacker but still outperforms competitors by a large margin. While attack detection is near 100% for all defenses, only Crossfire detects over 90% of bit flips across datasets and restores over 99% of the GNN's prediction quality for up to 55 bit flips. Moreover, Crossfire's reconstruction capabilities far exceed its weaker competitors. For example, for 15 bit flips, Crossfire achieves a 41% average reconstruction rate, well outperforming its closest competitor, NeuroPots, which only achieves 5%. This trend of superiority is maintained for up to 55 bit flips. Note that, while full recovery for large numbers of bit flips is modest (though significantly better than baselines), near-100% attack detection probabilities for fewer flips makes a BFA unlikely to cause undetected damage, as inducing 25 consecutive flips would already require hours (Wang et al. 2023a).

To formalize our findings, we conducted Welch's t-test to compare Crossfire's performance with that of NeuroPots and RADAR, testing the hypotheses formulated at the beginning of this section for $p < 10^{-3}$. For hypothesis **(a)**, we found a highly significant improvement in Crossfire's bit flip detection ratio ($t = 9.8, p = 10^{-18}$). For hypothesis **(b)**, Crossfire significantly outperformed its competitors in GNN reconstruction ratio ($t = 7.1, p = 2{\cdot}10^{-10}$). Regarding hypothesis **(c)**, Crossfire yielded a significantly smaller difference between pre-attack and post-reconstruction AP/AUROC ($t = 4.7, p = 5 \cdot 10^{-6}$). For hypothesis **(d)**, we found no significantly increased vulnerability in Crossfire-defended GNNs to the evaluated BFAs ($t = 2.8, p = 5 \cdot 10^{-3}$).

Overhead and Reliability For estimating relative computational overhead, we assume an `INT8` quantized simplistic message passing network operating over adjacency matrix $\mathbf{A}$, feature matrix $\mathbf{X}$ and weight matrix $\mathbf{W}$, computing $\mathbf{AXW}^T$. We evaluate for batch size 32 and a randomly connected graph with 5 or 10 nodes (i.e. $\mathbf{A} \in \{0,1\}^{(5\times5)}$ or $\mathbf{A} \in \{0,1\}^{(10\times10)}$). In practice, the number of nodes is higher (Table 1), so overhead can be assumed to be even lower. We run hashing sequentially to simulate the worst case of a system where parallelism is not possible. As shown in Figure 1, right, Crossfire requires one order of magnitude less computation time than even our simplistic GNN layer. The average runtime stays low even for large matrices and decreases relative to the GNN layer's computational demand as matrix sizes increase, demonstrating Crossfire's scalability. Comparing the CPU-based hashing with the `INT8` quantized GNN's complexity is appropriate, as `INT8` quantized models are typically run on CPUs on edge devices. The storage overhead scales effectively with increasing matrix sizes (Figure 1, left). Depending on the digest sizes and honeypot percentages, Crossfire was more efficient than NeuroPots and RADAR (Figure 1, center).

To test `Blake2b`'s reliability for post-attack verification, we induced 1, 5, and 10 random consecutive bit flips in uniformly initialized square `INT8` matrices of sizes ranging from 100 to 1000 (in increments of 100). We then tested the probability that a bit flip goes undetected for layer digest sizes of 1, 2, and 3. The experiment was repeated 100 times for each combination of the above parameters. We found that a digest size of 1 failed to detect 0.66%, 0.44%, and 0.22% of the 1, 5, and 10 bit flip sequences, respectively. However, any digest size ≥ 2 achieved a 100% detection rate, highlighting `Blake2b`'s usefulness for integrity verification.

Conclusion

We introduced Crossfire, the first defense framework designed to protect GNNs from BFAs which is robust and retraining-free. Tested on six datasets accross 2,160 experiments, Crossfire surpasses existing methods in both detection and recovery performance. Crossfire achieves near-perfect attack and bit-flip detection and typically restores prediction quality to pre-attack levels. On average, Crossfire offers a 21.8% higher probability of full reconstruction and improves post-repair prediction quality by 10.85% over its competitors. Crossfire's computational and storage overhead is negligible compared to the inherent complexity of the simplest GNNs. The innovative use of hashing and honeypots, combined with leveraging sparsity and the systematic unflipping of bits in OOD weights, enables identification and repair of perturbed elements with minimal overhead. Crossfire addresses a critical gap in GNN security, offering a scalable solution to safeguard GNNs in adversarial scenarios.

Acknowledgments

This work was supported by the Vienna Science and Technology Fund (WWTF) [10.47379/VRG19009].

References

ARM, A. 2009. Security technology building a secure system using trustzone technology (white paper). *ARM Limited*.

Aumasson, J.-P.; Neves, S.; Wilcox-O'Hearn, Z.; and Winnerlein, C. 2013. BLAKE2: simpler, smaller, fast as MD5. In *Applied Cryptography and Network Security: 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings 11*, 119–135. Springer.

Batina, L.; Bhasin, S.; Jap, D.; and Picek, S. 2018. CSI neural network: Using side-channels to recover your artificial neural network information. *CoRR*, abs/2204.07697.

Bengio, Y.; Léonard, N.; and Courville, A. C. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *CoRR*, abs/1308.3432.

Breier, J.; Hou, X.; Jap, D.; Ma, L.; Bhasin, S.; and Liu, Y. 2018. Practical Fault Attack on Deep Neural Networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, 2204–2206. New York, NY, USA: Association for Computing Machinery.

Cheung, M.; and Moura, J. M. 2020. Graph neural networks for covid-19 drug discovery. In *2020 IEEE International Conference on Big Data (Big Data)*, 5646–5648. IEEE.

Chitsaz, K.; Mordido, G.; David, J.-P.; and Leduc-Primeau, F. 2023. Training DNNs Resilient to Adversarial and Random Bit-Flips by Learning Quantization Ranges. *Transactions on Machine Learning Research*.

Dai, E.; Zhao, T.; Zhu, H.; Xu, J.; Guo, Z.; Liu, H.; Tang, J.; and Wang, S. 2022. A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability. *CoRR*, abs/2204.08570.

Fey, M.; and Lenssen, J. E. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Frankle, J.; and Carbin, M. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *CoRR*, abs/1803.03635.

Gao, Y.; Hasegawa, H.; Yamaguchi, Y.; and Shimada, H. 2022. Malware Detection by Control-Flow Graph Level Representation Learning With Graph Isomorphism Network. *IEEE Access*, 10: 111830–111841.

Guo, J.; Karpman, P.; Nikolić, I.; Wang, L.; and Wu, S. 2014. Analysis of BLAKE2. In *Topics in Cryptology–CT-RSA 2014: The Cryptographer's Track at the RSA Conference 2014, San Francisco, CA, USA, February 25-28, 2014. Proceedings*, 402–423. Springer.

He, Z.; Rakin, A. S.; Li, J.; Chakrabarti, C.; and Fan, D. 2020. Defending and Harnessing the Bit-Flip Based Adversarial Weight Attack. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 14083–14091.

Hector, K.; Moëllic, P.-A.; Dumont, M.; and Dutertre, J.-M. 2022. A Closer Look at Evaluating the Bit-Flip Attack Against Deep Neural Networks. In *2022 IEEE 28th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 1–5.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33: 22118–22133.

Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; and Kalenichenko, D. 2018. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2704–2713.

Javaheripi, M.; and Koushanfar, F. 2021. HASHTAG: Hash Signatures for Online Detection of Fault-Injection Attacks on Deep Neural Networks. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 1–9.

Jin, W.; Li, Y.; Xu, H.; Wang, Y.; Ji, S.; Aggarwal, C.; and Tang, J. 2021. Adversarial attacks and defenses on graphs. *ACM SIGKDD Explorations Newsletter*, 22(2): 19–34.

Khare, Y.; Lakara, K.; Inukonda, M. S.; Mittal, S.; Chandra, M.; and Kaushik, A. 2022. Design and Analysis of Novel Bit-flip Attacks and Defense Strategies for DNNs. In *2022 IEEE Conference on Dependable and Secure Computing (DSC)*, 1–8.

Kullback, S.; and Leibler, R. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22: 79–86.

Kummer, L.; Moustafa, S.; Kriege, N.; and Gansterer, W. 2024. Attacking Graph Neural Networks with Bit Flips: Weisfeiler and Lehman Go Indifferent. *ACM SIGKDD '24 (Accepted Preprint, CoRR, abs/2311.01205)*.

Kummer, L.; Sidak, K.; Reichmann, T.; and Gansterer, W. 2023. Adaptive Precision Training (AdaPT): A dynamic quantized training approach for DNNs. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*, 559–567. SIAM.

Li, J.; Rakin, A. S.; He, Z.; Fan, D.; and Chakrabarti, C. 2021. RADAR: Run-time Adversarial Weight Attack Detection and Accuracy Recovery. In *2021 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 790–795.

Li, J.; Rakin, A. S.; Xiong, Y.; Chang, L.; He, Z.; Fan, D.; and Chakrabarti, C. 2020a. Defending Bit-Flip Attack through DNN Weight Reconstruction. *DAC*.

Li, Y.; Qian, B.; Zhang, X.; and Liu, H. 2020b. Graph neural network-based diagnosis prediction. *Big Data*.

Lipp, M.; Schwarz, M.; Raab, L.; Lamster, L.; Aga, M. T.; Maurice, C.; and Gruss, D. 2020. Nethammer: Inducing rowhammer faults through network requests. *EuroS&PW*.

Liu, M.; Gao, H.; and Ji, S. 2020. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 338–348.

Liu, Q.; Yin, J.; Wen, W.; Yang, C.; and Sha, S. 2023. NeuroPots: Realtime Proactive Defense against Bit-Flip Attacks in Neural Networks. *pre-print accepted at USENIX Security*.

Liu, Z.; Li, X.; Peng, H.; He, L.; and Philip, S. Y. 2020. Heterogeneous similarity graph neural network on electronic health records. *Big Data*.

Lu, H.; and Uddin, S. 2021. A weighted patient network-based framework for predicting chronic diseases using graph neural networks. *Scientific reports*, 11(1): 22607.

Ma, J.; Ding, S.; and Mei, Q. 2020. Towards More Practical Adversarial Attacks on Graph Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 33, 4756–4766. Curran Associates, Inc.

Malach, E.; Yehudai, G.; Shalev-Schwartz, S.; and Shamir, O. 2020. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, 6682–6691. PMLR.

McKeen, F.; Alexandrovich, I.; Anati, I.; Caspi, D.; Johnson, S.; Leslie-Hurd, R.; and Rozas, C. 2016. Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave. In *Proceedings of the Hardware and Architectural Support for Security and Privacy 2016*, 1–9.

Morris, C.; Fey, M.; and Kriege, N. 2021. The Power of the Weisfeiler-Leman Algorithm for Machine Learning with Graphs. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 4543–4550. International Joint Conferences on Artificial Intelligence Organization.

Mutlu, O.; and Kim, J. S. 2019. Rowhammer: A retrospective. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(8): 1555–1571.

Özdenizci, O.; and Legenstein, R. 2022. Improving Robustness Against Stealthy Weight Bit-Flip Attacks by Output Code Matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 13388–13397.

Pinto, S.; and Santos, N. 2019. Demystifying arm trustzone: A comprehensive survey. *ACM computing surveys (CSUR)*, 51(6): 1–36.

Qian, C.; Zhang, M.; Nie, Y.; Lu, S.; and Cao, H. 2023. A Survey of Bit-Flip Attacks on Deep Neural Network and Corresponding Defense Methods. *Electronics*, 12(4): 853.

Rakin, A. S.; He, Z.; and Fan, D. 2019. Bit-Flip Attack: Crushing Neural Network With Progressive Bit Search. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 1211–1220.

Rao, V.; and Prema, K. V. 2019. Comparative Study of Lightweight Hashing Functions for Resource Constrained Devices of IoT. In *2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*, 1–5.

Sugier, J. 2017. Memory resources in hardware implementations of BLAKE and BLAKE2 hash algorithms. *Journal of Polish Safety and Reliability Association*, 8(1).

Sun, Y.; Wang, S.; Tang, X.; Hsieh, T.-Y.; and Honavar, V. 2020. Adversarial Attacks on Graph Neural Networks via Node Injections: A Hierarchical Reinforcement Learning Approach. 673–683.

Sun, Z.; Yin, H.; Chen, H.; Chen, T.; Cui, L.; and Yang, F. 2021. Disease Prediction via Graph Neural Networks. *IEEE Journal of Biomedical and Health Informatics*, 25(3): 818–826.

Wang, J.; Zhang, Z.; Wang, M.; Qiu, H.; Zhang, T.; Li, Q.; Li, Z.; Wei, T.; and Zhang, C. 2023a. Aegis: Mitigating Targeted Bit-flip Attacks against Deep Neural Networks. In *32nd USENIX Security Symposium (USENIX Security 23)*, 2329–2346.

Wang, Z.; Lin, Z.; Li, S.; Wang, Y.; Zhong, W.; Wang, X.; and Xin, J. 2023b. Dynamic Multi-Task Graph Isomorphism Network for Classification of Alzheimer's Disease. *Applied Sciences*, 13(14): 8433.

Welling, M.; and Kipf, T. N. 2016. Semi-supervised classification with graph convolutional networks. In *J. International Conference on Learning Representations (ICLR 2017)*.

Wu, L.; Cui, P.; Pei, J.; Zhao, L.; and Song, L. 2022. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer.

Wu, Z.; Ramsundar, B.; Feinberg, E. N.; Gomes, J.; Geniesse, C.; Pappu, A. S.; Leswing, K.; and Pande, V. 2018. MoleculeNet: a benchmark for molecular machine learning. *Chemical science*, 9(2): 513–530.

Xiang, L.; Zeng, X.; Niu, Y.; and Liu, Y. 2019. Study of Sensitivity to Weight Perturbation for Convolution Neural Network. *IEEE Access*, 7: 93898–93908.

Xiaoqin Zeng; and Yeung, D. S. 2001. Sensitivity analysis of multilayer perceptron to input and weight perturbations. *IEEE Transactions on Neural Networks*, 12(6): 1358–1366.

Xiong, J.; Xiong, Z.; Chen, K.; Jiang, H.; and Zheng, M. 2021. Graph neural networks for automated de novo drug design. *Drug Discovery Today*, 26(6): 1382–1393.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *7th International Conference on Learning Representations, ICLR*.

Yan, M.; Fletcher, C.; and Torrellas, J. 2020. Cache telepathy: Leveraging shared resource attacks to learn DNN architectures. In *USENIX Security Symposium*.

Yao, F.; Rakin, A. S.; and Fan, D. 2020. Deephammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips. In *29th $USENIX$ Security Symposium ($USENIX$ Security 20)*.

Zhang, S.; Chen, H.; Sun, X.; Li, Y.; and Xu, G. 2022a. Unsupervised graph poisoning attack via contrastive loss back-propagation. In *Proceedings of the ACM Web Conference 2022*, 1322–1330.

Zhang, Z.; Wang, M.; Chen, W.; Qiu, H.; and Qiu, M. 2022b. Mitigating Targeted Bit-Flip Attacks via Data Augmentation: An Empirical Study. In *Knowledge Science, Engineering and Management: 15th International Conference, KSEM 2022*.

A.3 On the Relationship Between Robustness and Expressivity

Title	On the Relationship Between Robustness and Expressivity of Graph Neural Networks
Authors	Lorenz Kummer, Samir Moustafa, Wilfried Gansterer, Nils Kriege
Venue	Proceedings of The 28th International Conference on Artificial Intelligence and Statistics (CORE ranking: A)
Identifier	`ISSN:2640-3498`
Division of work	Lorenz Kummer: Conceived the initial idea; formulated the problem and theoretical derivations; designed and implemented the experiments; conducted the evaluation; wrote the article and created visualizations; produced the conference poster; reviewed the related work. Wilfried Gansterer: Participated in discussions; reviewed and refined the article and conference poster; provided mentoring and supervision. Nils Kriege: Participated in discussions; reviewed and refined the article – particularly its theoretical underpinnings – and the conference poster; provided mentoring and supervision.

On the Relationship Between Robustness and Expressivity of Graph Neural Networks

Lorenz Kummer[1,2]
[1]Faculty of Computer Science
University of Vienna
Vienna, Austria

Wilfried N. Gansterer[1]
[2]Doctoral School Computer Science
University of Vienna
Vienna, Austria

Nils M. Kriege[1,3]
[3]Research Network Data Science
University of Vienna
Vienna, Austria

Abstract

We investigate the vulnerability of Graph Neural Networks (GNNs) to bit-flip attacks (BFAs) by introducing an analytical framework to study the influence of architectural features, graph properties, and their interaction. The expressivity of GNNs refers to their ability to distinguish non-isomorphic graphs and depends on the encoding of node neighborhoods. We examine the vulnerability of neural multiset functions commonly used for this purpose and establish formal criteria to characterize a GNN's susceptibility to losing expressivity due to BFAs. This enables an analysis of the impact of homophily, graph structural variety, feature encoding, and activation functions on GNN robustness. We derive theoretical bounds for the number of bit flips required to degrade GNN expressivity on a dataset, identifying ReLU-activated GNNs operating on highly homophilous graphs with low-dimensional or one-hot encoded features as particularly susceptible. Empirical results using ten real-world datasets confirm the statistical significance of our key theoretical insights and offer actionable results to mitigate BFA risks in expressivity-critical applications.

1 INTRODUCTION

Graph Neural Networks (GNNs) have recently emerged as a dominant technique for machine learning on structured data. Complex objects such as molecules, proteins, or social networks can be represented as graphs, with nodes as components and edges as their relations, both potentially carrying features. By generalizing deep learning techniques to graphs, GNNs open up new domains, including the analysis of financial and social networks, medical data, as well as chem- and bioinformatics (Lu and Uddin, 2021; Sun et al., 2021; Gao et al., 2022; Wu et al., 2018; Xiong et al., 2021). Their increasing adoption across various fields necessitates examining robustness and potential security issues. Traditionally, adversarial attacks on GNNs target input graph data (Wu et al., 2022), aiming to induce perturbations before training (*poisoning*) to produce faulty models (Ma et al., 2020; Wu et al., 2022), or during inference (*evasion attacks*) to degrade prediction quality. These perturbations are achieved by modifying node features, adding or deleting edges, or injecting new nodes (Sun et al., 2020; Wu et al., 2022). An overview and classification of existing attacks and corresponding defenses, along with representative algorithms, is provided by Jin et al. (2021).

A novel paradigm for attacking GNNs is bit-flip attacks (BFAs). In contrast to the orthogonal approaches of poisoning and evasion, BFAs degrade the model directly by manipulating the binary in-memory representations of the model's trainable parameters. The feasibility of BFAs, which are typically executed at inference time, is well-documented (Mutlu and Kim, 2019; Yao et al., 2020; Breier et al., 2018). However, unlike Convolutional Neural Networks (CNNs), where the security issues of BFAs are thoroughly studied (Qian et al., 2023; Khare et al., 2022) and largely attributed to weight sharing in convolutional filters (Hector et al., 2022), the robustness of GNNs, which lack such filters, remains underexplored, despite their use in safety-critical domains such as medical diagnoses (Gao et al., 2022; Lu and Uddin, 2021), electronic health record modeling (Sun et al., 2021), and drug development (Xiong et al., 2021). To date, random (Jiao et al., 2022) and gradient-based (Wu et al., 2023; Kummer et al., 2024) BFAs on GNNs have primarily been studied empirically. The intricate mechanisms determining GNN

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

vulnerability and their relation to learning tasks, graph properties and key properties such as expressivity remain poorly understood theoretically. Expressivity in particular is a fundamental factor, even if in practice, other factors that go beyond expressivity influence predictive performance as well. Any GNN solving a certain node classification task must at least pairwise distinguish those nodes associated with different classes. This equates to such nodes being mapped to embedding vectors of which at least one element is different, reflecting (for a message passing based GNN with k layers) that the two nodes' unfolding trees of height k are non-isomorphic. Likewise, any model solving a certain graph classification task must at least pairwise distinguish those non-isomorphic graphs belonging to different classes, implying that for each pair the set of node embeddings must be different in at least one element, linking again to the unfolding tree isomorphism problem and therefore expressivity.

Related Work Research on GNN resilience to bit flips is scarce: Jiao et al. (2022) studied random bit faults in floating-point hardware and observed that resilience varies significantly across models and datasets, but without providing a theoretical explanation. Focusing on detecting bit flips in GNN weights, Wu et al. (2023) transferred the seminal BFA of Rakin et al. (2019) from quantized CNNs to GNNs, modifying it to flip bits in the exponent of a weight's floating-point representation. Amir et al. (2024)'s results indicate that GNN expressivity is generally resilient to random bit flips. Thus, Kummer et al. (2024) proposed a BFA targeting quantized GNNs, exploiting their expressivity to degrade classification accuracy by targeting injectivity in neighborhood aggregation. Given the key role of expressivity in GNNs, it is crucial to better understand its relationship to robustness against BFAs.

Contribution We characterize the properties of bit flips degrading GNN expressivity and establish formal bounds on the number of bit flips required. We first analyze general vulnerabilities of neural multiset functions based on element-wise summation and then focus on derived GNNs, where instances like Graph Isomorphism Network (GIN) achieve an expressivity equivalent to the Weisfeiler-Leman test (Xu et al., 2019). We show that a GNN's vulnerability to expressivity degradation from bit flips depends on parameter count, number of bits used to represent a value (bit width), diversity of distinguishable subgraphs, variety of GNN embeddings, input feature encoding, dataset homophily, activation function, and the bit's location in the numerical representation.

1.1 Preliminaries

A *graph* G is a pair (V, E) of a finite set of *nodes* V and *edges* $E \subseteq \{\{u, v\} \subseteq V\}$. The set of nodes and edges of G is denoted by $V(G)$ and $E(G)$, respectively. The *neighborhood* of v in $V(G)$ is $N(v) = \{u \in V(G) \mid \{u, v\} \in E(G)\}$. If there exists a bijection $\varphi\colon V(G) \to V(H)$ with $\{u, v\}$ in $E(G)$ if and only if $\{\varphi(u), \varphi(v)\}$ in $E(H)$ for all u, v in $V(G)$, we call the two graphs G and H *isomorphic* and write $G \simeq H$. For two graphs with roots $r \in V(G)$ and $r' \in V(H)$, the bijection must additionally satisfy $\varphi(r) = r'$. The equivalence classes induced by $\simeq$ are referred to as *isomorphism types*. A *node colored* or *labeled graph* is a pair (G, l) consisting of a graph G endowed with a *node coloring* $l\colon V(G) \to \Sigma$. We call $l(v)$ a *label* or *color* of $v \in V(G)$. We denote a multiset by $\{\!\{\ldots\}\!\}$.

The Weisfeiler-Leman Algorithm Let (G, l) denote a labelled graph. In every iteration $t > 0$, a node coloring $c_l^{(t)}\colon V(G) \to \Sigma$ is computed, which depends on the coloring $c_l^{(t-1)}$ of the previous iteration. At the beginning, the coloring is initialized as $c_l^{(0)} = l$. In subsequent iterations $t > 0$, the coloring is updated according to $c_l^{(t)}(v) = \textsc{Hash}\left(c_l^{(t-1)}(v), \{\!\{c_l^{(t-1)}(u) | u \in N(v)\}\!\}\right)$, where HASH is an injective mapping of the above pair to a unique value in Σ, that has not been used in previous iterations. The HASH function can, for example, be realized by assigning new consecutive integer values to pairs when they occur for the first time (Shervashidze et al., 2011). Let $C_l^{(t)}(G) = \{\!\{c_l^{(t)}(v) \mid v \in V(G)\}\!\}$ be the multiset of colors a graph exhibits in iteration t. The iterative coloring terminates if $|C_l^{(t-1)}(G)| = |C_l^{(t)}(G)|$, i.e., the number of colors does not change between two iterations. For testing whether two graphs G and H are isomorphic, the above algorithm is run in parallel on both G and H. If $C_l^{(t)}(G) \neq C_l^{(t)}(H)$ for any $t \geq 0$, then G and H are not isomorphic. The label $c_l^{(t)}(v)$ assigned to a node v in the tth iteration of the 1-WL test encodes the isomorphism type of the tree of height t representing the t-hop neighborhood of v, see (D'Inverno et al., 2021; Jegelka, 2022; Schulz et al., 2022) for details.

From DeepSets to Graph Neural Networks DeepSets (Zaheer et al., 2017) is an architecture for modeling set functions, suitable for tasks with unordered data. Given a set $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, a deep set function is of the form $f(X) = \rho\left(\sum_{i=1}^{k} \phi(\mathbf{x})\right)$. Due to the summation, f is invariant to the permutation of the input elements, capturing the intrinsic property of sets. The function ϕ maps each element of X to a feature space, in which their sum is calculated and then transformed by ρ into the final out-

put. Typically both functions are realized by a Multi-Layer Perceptron (MLP), whose universal approximation property leads to universal approximators of set functions (Zaheer et al., 2017), albeit with certain practical caveats (Wagstaff et al., 2022). This concept extends to multisets, reflecting both identity and multiplicity of elements (Xu et al., 2019; Amir et al., 2024).

In this context, GNNs can be seen as stacked neural (multi)set functions, where each layer aggregates and combines node features to reflect the multiset nature of graph neighborhoods, a technique also known as *message-passing* (MP). Specifically, a *neural moment* function $\hat{f}$ derived from sum-pooling operations is given by $\hat{f}(\{\!\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}\!\}) = \sum_{i=1}^{k} f(\mathbf{x}_i)$, where $f\colon V^d \to V^m$ is an MLP mapping elements from a set V^d to a vector space V^m (Amir et al., 2024). This ensures that for a suitable choice of d and m and parameterization of the MLP, each multiset is uniquely represented by its moment, allowing distinct multisets to have distinct representations, as in DeepSets. GNNs apply this principle by using an MLP in each layer's update rule,

$$\mathbf{h}_v^{(k)} = \mathrm{MLP}^{(k)}\left(\mathbf{h}_v^{(k-1)} + \sum_{u \in N(v)} \mathbf{h}_u^{(k-1)}\right), \quad (1)$$

where $\mathbf{h}_v^{(k)}$ represents the embedding of node v at layer k. This effectively extends DeepSets to graph-structured data, as the layer-wise combination of node features from a node and its neighborhood allows such a model to learn to distinguish certain graph structures. If the input features are one-hot encodings, this ensures injectivity of the summation at the first layer, even without an initial MLP. For graph-level tasks, the readout function typically aggregates the outputs across all layers to obtain a graph-level embedding $\mathbf{h}_G$, where $\|$ denotes concatenation:

$$\mathbf{h}_G = \Big\|_{k=0}^{n} \sum_{v \in V(G)} \mathbf{h}_v^{(k)}. \quad (2)$$

The update rule can also be expressed in matrix form using the adjacency matrix $\mathbf{A}$ and node feature matrix $\mathbf{H}^{(k)} = \mathrm{MLP}^{(k)}\left((\mathbf{A} + \mathbf{I}) \cdot \mathbf{H}^{(k-1)}\right)$, which allows for an easier notation of a GNN as $\Phi^{(k)}\colon D \to Q^{m_k}$, mapping domain $D = \{(\mathbf{A}_1, \mathbf{X}_1), \ldots (\mathbf{A}_n, \mathbf{X}_n)\}$ to some codomain Q^{m_k} with $\mathbf{H}^{(0)} = \mathbf{X}$. For node features $\mathbf{X}$ representing encodings of the graphs labels, the set $D = \{(\mathbf{A}_1, \mathbf{X}_1), \ldots (\mathbf{A}_n, \mathbf{X}_n)\}$ is only a different notation for the set of labeled graphs $\{(G_1, l_1), \ldots (G_n, l_n)\}$. Hence, we do not distinguish between the two in the following, as it will be clear from the context which one is used.

Graph Isomorphism Network (GIN) (Xu et al., 2019) extends this approach, achieving expressivity provably equal to 1-WL. GIN's update rule differs from Equation (1) (and its matrix notation) by multiplying the embeddings $\mathbf{h}_v^{(k-1)}$ by a learnable parameter $(1 + \epsilon^{(k)})$, differentiating a node's own features from aggregated neighbor features. While much research aims to surpass the expressivity of GIN and the 1-WL test (Morris et al., 2023), neighborhood aggregation remains prevalent, and 1-WL distinguishes most graphs in common benchmarks (Zopf, 2022; Morris et al., 2021).

2 BIT FLIPS AND EXPRESSIVITY

We first examine the general properties and vulnerabilities of neural moments, followed by a detailed analysis of neural moment-based GNN architectures.

Let Ω_n^d be the set of all bounded multisets with at most $n \in \mathbb{N}_{\geq 1}$ elements from some $d \in \mathbb{N}_{\geq 2}$ dimensional vector space V^d. For an $X \in \Omega_n^d$, we denote as $m_X(\mathbf{x})$ the multiplicity function returning the number of occurrences of $\mathbf{x} \in X$ and as $S(X)$ the support set of X, the set of X's distinct elements. Further, let the moment $\hat{f} : \Omega_n^d \to V^m$ induced by some function $f : V^d \to V^m$ be defined as $\hat{f}(X) = \sum_{\mathbf{x} \in X} f(\mathbf{x})$ (Amir et al., 2024). We now explore the relationship between the pre-aggregation linearly independent mapping of distinct multiset elements by f, the injectivity of f and moment injectivity of $\hat{f}$.

A sufficient condition for f to be injective is that for any multiset $A \in \Omega_n^d$, the vectors $\{\!\{f(\mathbf{x}) \mid \mathbf{x} \in S(A)\}\!\}$ are linearly independent. Next, we show that for the set of bounded multisets Ω_n^d with $n \geq 2$, a pre-aggregation linearly independent mapping of distinct multiset elements by f implies moment injectivity (i.e. the injective mapping of distinct multisets) of $\hat{f}$. We chose $n \geq 2$ because for $n = 1$, the linear independence of the pre-aggregation mapping of distinct multiset elements is always trivially given since each multiset has at most one element.

Proposition 1 (Sufficient condition for moment injectivity)**.** *Let $\hat{f} : \Omega_n^d \to V^m$ denote the moment induced by any $f : V^d \to V^m$. Then for $n \geq 2$, the second statement follows from the first:*

1. For all $A \in \Omega_n^d$, if $\sum_{\mathbf{x} \in S(A)} f(\mathbf{x})\lambda_{\mathbf{x}} = \mathbf{0}, \quad \lambda_{\mathbf{x}} \in V$, then $\lambda_{\mathbf{x}} = 0$ for all $\mathbf{x} \in A$.
2. For all $A, B \in \Omega_n^d$ $\hat{f}(A) = \hat{f}(B)$ implies $A = B$.

To summarize, non-injectivity of f implies, in particular, linear dependence of the elements $\{\!\{f(\mathbf{x}), \mathbf{x} \in A\}\!\}$. Proposition 1 guarantees moment injectivity of $\hat{f}$ only under linear independence. Thus, without linear independence, moment injectivity of $\hat{f}$ is not guaranteed, and we cannot exclude the possibility that $\hat{f}$ fails to distinguish distinct multisets. This is consistent with

the finding that projection into a low rank sub-space is the underlying cause of indistinguishable node embeddings in GNNs (also known as over-smoothing) (Roth and Liebig, 2024) as well as previous conjectures on GNN vulnerability to bit flips being related to the injectivity of the MLP's modelling f (Kummer et al., 2024). Furthermore, it has profound implications for GNNs such as GIN, which rely on moment-injectivity by modeling f using MLP's, guaranteeing their ability to distinguish multisets representing 1-WL subtrees.

We now investigate the relation between bit flips in the MLPs of neural moment-based GNNs and their expressivity on a certain, bounded graph dataset D. To this purpose, we first formulate a well-known result for easy reference later on. Let $\Phi^{(k)}$ be a GNN with k layers, as in Equation (1). Then, $\text{MLP}_l^{(j)}, 1 \leq j \leq k$ denotes the l-layer MLP of $\Phi^{(k)}$'s jth layer. Let the ith layer of $\text{MLP}_l^{(j)}$ be $\sigma \circ \mathbf{W}^{(j,i)} : Q^{n_{j,i}} \to Q^{m_{j,i}}$, $1 \leq i \leq l$. Clearly, if all layers $\sigma \circ \mathbf{W}^{(j,i)}$ of an $\text{MLP}_l^{(j)}$ are injective, the $\text{MLP}_l^{(j)}$ is injective and the following holds.

Lemma 1 (Sufficient condition for the 1-WL expressivity of $\Phi^{(k)}$ (Xu et al., 2019))**.** *If all layers* $\text{MLP}_l^{(j)}$*, for* $1 \leq j \leq k$*, of the GNN* $\Phi^{(k)}$ *are injective and each layer's aggregation rule can distinguish a node's own features and the aggregated features from its neighbors, then* $\Phi^{(k)}$ *is as expressive as the 1-WL test.*

Note that Lemma 1 describes a sufficient condition. That is, a GNN might be maximally expressive on a certain, bounded dataset D even if not all of it's $\text{MLP}_l^{(j)}$'s are injective (Kummer et al., 2024).

Generalizing (Puthawala et al., 2022), we now define a general condition for the injectivity of $\sigma \circ \mathbf{W}^{(j,i)} : Q^{n_{j,i}} \to Q^{m_{j,i}}$ with linear transformation $\mathbf{W}^{(j,i)} \in U^{m_{j,i} \times n_{j,i}}$, $1 < n_{j,i} \leq m_{j,i}$, $Q \subset U$. Note the distinction between Q and U, which we detail below. The domain $Q^{n_{j,0}}$ is defined as the set of all possible aggregates $\text{MLP}_l^{(j)}$ might receive on a given set D of graphs after j layers. Thus, $Q^{n_{j,0}}$'s cardinality is bounded by the number of 1-WL colors occurring for any graph $G \in D$ after j iterations $|Q^{n_{j,0}}| \leq |\{c_l^{(j)}(v) \mid v \in V(G), G \in D\}|$. Since any layer of $\text{MLP}_l^{(j)}$ can produce at most as many distinguishable outputs as there are distinct inputs, this also holds true for all $0 < i \leq l$. This implies that, likewise, the codomain $Q^{m_{j,i}}$'s cardinality is constrained to the cardinality of the set of the possible transformed aggregates. These constraints on the vector spaces $Q^{n_{j,i}}$ and $Q^{m_{j,i}}$ also bring limitations on the underlying field Q, which is implicitly constrained in its elements. These constraints do not apply to the vector space of $\mathbf{W}^{(j,i)} \in U^{m_{j,i} \times n_{j,i}}$, $1 < n_{j,i}$, which is why we define Q as subset of U.

Further, in the following, we assume σ to be an analytic injective activation function unless stated otherwise. For such an activation function, a layer $\sigma \circ \mathbf{W}^{(j,i)}$ can be globally injective without the requirements on $\mathbf{W}^{(j,i)}$'s width and $U^{m_{j,i} \times n_{j,i}}$ applying to piece-wise-linear (PwL) activations (Amir et al., 2024) such as ReLU (Puthawala et al., 2022). We will discuss the special case of ReLU as an instance of PwL activations later in Section 2.3.

Lemma 2 (Equivalence of conditions for layer injectivity)**.** *Let* $\mathbf{W}^{(j,i)} \in U^{m_{j,i} \times n_{j,i}}$ *be a linear transformation with* $1 < n_{j,i} \leq m_{j,i}$*, and let* $R_{\mathbf{W}^{(j,i)}} = \{\mathbf{w}_r\}_{r=0}^{m_{j,i}}$ *denote the set of row vectors of* $\mathbf{W}^{(j,i)}$*. Consider an injective elementwise activation function* $\sigma : U \to Q$*. Then, the function* $\sigma \circ \mathbf{W}^{(j,i)} : Q^{n_{j,i}} \to Q^{m_{j,i}}$ *is injective if and only if, for any distinct* $\mathbf{x}_u, \mathbf{x}_v \in Q^{n_{j,i}}$*, there exists a row vector* $\mathbf{w}_r \in R_{\mathbf{W}^{(j,i)}}$ *such that* $\langle \mathbf{x}_u, \mathbf{w}_r \rangle \neq \langle \mathbf{x}_v, \mathbf{w}_r \rangle$*.*

If $\mathbf{W}^{(j,i)}$ is of full rank, then $\sigma \circ \mathbf{W}^{(j,i)} : Q^{n_{j,i}} \to Q^{m_{j,i}}$ satisfies the injectivity conditions outlined in Lemma 2. The converse does not need to be true, though. Since the domain $Q^{n_{j,i}}$ is constrained by the number of possible aggregates at layer (j, i), an injective mapping of $Q^{n_{j,i}}$ to $Q^{m_{j,i}}$ could also be facilitated if $\mathbf{W}^{(j,i)}$ does not have full rank.

We now provide the definition of a GNN that satisfies the outlined criteria for injectivity on a constrained domain and codomain.

Definition 1 (Maximally expressive GNN)**.** *A GNN* $\Phi^{(k)}$ *is* maximally expressive *if, for every layer of each* $\text{MLP}_l^{(j)}$ *in* $\Phi^{(k)}$*, the function* $\sigma \circ \mathbf{W}^{(j,i)}$ *satisfies the injectivity conditions outlined in Lemma 2 and its aggregation rule can distinguish a node's own features and the aggregated features from its neighbors. Consequently, by Lemma 1,* $\Phi^{(k)}$ *is as expressive as 1-WL.*

Although the subsequent analysis of node and graph-level expressivity (Sections 2.1 and 2.2) centers on neural moment-based GNNs such as GIN, it readily extends to other architectures. This includes Graph Convolutional Networks (Welling and Kipf, 2016) (GCN), by modeling GCN's transformations as 1-layer MLPs, which we include in our empirical evaluation, but (except for some of the special cases we discuss in Section 2.3) also translates to, e.g., Graph Attention Networks (Veličković et al., 2018) (GATs), which are likewise neural moment-based architectures, even if their sum is weighted by dynamically computed edge weights: their expressivity in the sense of their ability to distinguish non-isomorphic graphs is likewise bounded by the 1-WL algorithm and likewise intrinsically connected to the injectivity of the functions approximated by their trainable parameters (i.e. linear transformations combined with non-linear activations).

2.1 Node Level Expressivity

We explore the node-level expressivity of GNN $\Phi^{(k)}$, specifically lower and upper bounds on the number of bit flips in the MLPs required to prevent a maximally expressive GNN from distinguishing all pairs of nodes in all graphs G of a dataset D that 1-WL distinguishes. The failure to distinguish even a single pair of WL-distinguishable nodes u, v of any $G \in D$ suffices to break node-level 1-WL expressivity. In special cases, this could occur with a single bit flip: without constraints on node features or structures, inputs (aggregates) $\mathbf{x}_u, \mathbf{x}_v$ for nodes u, v at layer (j, i) may differ by only one element. If only one row $\mathbf{w}_r$ of $\mathbf{W}^{(j,i)}$ satisfies $\langle \mathbf{x}_u, \mathbf{w}_r \rangle \neq \langle \mathbf{x}_v, \mathbf{w}_r \rangle$, a single bit flip could map the distinguishing element to an identical value in the dot product, causing $\langle \mathbf{x}_u, \mathbf{w}_r \rangle = \langle \mathbf{x}_v, \mathbf{w}_r \rangle$ and further post-activation equality. By Lemmas 1 and 2, maximal expressivity of $\Phi^{(k)}$ as in Definition 1 is no longer guaranteed.

Of greater interest is constructing a general upper bound. This can be achieved by selecting nodes u, v at layer (j, i) with the most distinctive aggregates $\mathbf{x}_u, \mathbf{x}_v$ among all nodes in the joint set $V_\cup(D) = \bigcup_{G \in D} V(G)$. Note that the loss of distinguishability between $\mathbf{x}_u$ and $\mathbf{x}_v$ post-MLP_l^j's application implies a loss of MLP_l^j's injectivity, and thus, by Proposition 1, the potential loss of layer l's moment injectivity. In neural moment-based architectures, losing moment injectivity can prevent the distinction of multisets and thus node neighborhoods, which is required for node-level expressivity.

Theorem 1 (Node-level upper bound)**.** *Let $\Phi^{(k)}(\mathbf{A}, \mathbf{X})\colon D \to Q^{m_k}$ be a maximally expressive GNN. The node-level expressivity of $\Phi^{(k)}$ can be compromised by $\mathcal{O}(d_{j,i} \cdot m_{j,i} \cdot b)$ bit flips, where $d_{j,i} = \max_{\mathbf{x}_u, \mathbf{x}_v \in Q^{n_{j,i}}} \|\mathbf{x}_u - \mathbf{x}_v\|_0$ for $u, v \in V_\cup(D)$ and b is the bit width.*

Theorem 1, for which we provide computational complexity estimates in the supplementary material, effectively states that in the general case – without constraints on graph structure or input features – all neurons in a layer associated with differing aggregate elements must be zeroed out. While cases can be constructed where fewer bit flips suffice to negate the sufficient condition of $\mathbf{W}^{(j,i)}$ having full rank, this assumption does not hold in a worst-case analysis. Moreover, if $\max_{\mathbf{x}_u, \mathbf{x}_v \in Q^{n_{j,i}}} \|\mathbf{x}_u - \mathbf{x}_v\|_0 = n_{j,i}$, then $\mathcal{O}(d_{j,i} \cdot m_{j,i} \cdot b)$ degenerates to $\mathcal{O}(n_{j,i} \cdot m_{j,i} \cdot b)$, implying that distinguishability at layer (j, i) is robust up to zeroing out every neuron in $\mathbf{W}^{(j,i)}$ for nodes u and v. However, we can set limits on both node embedding diversity and variability, as shown in our analysis of graph-level expressivity and specific cases.

2.2 Graph Level Expressivity

For graph-level expressivity, we explore bounds on the number of bit flips required to prevent the GNN from distinguishing two graphs that are distinguishable by 1-WL. Similar to node-level expressivity, we say $\Phi^{(k)}$ is maximally expressive for a dataset D if it can distinguish any two graphs $G \not\simeq H$ that 1-WL distinguishes. Two graphs are distinguished by 1-WL if they have different colors or color multiplicities after t iterations, meaning a GNN distinguishes them if they receive at least one differing node embedding or different embedding multiplicities. Thus, we examine the bit flips needed to align embeddings of non-isomorphic graphs. Since color (embedding) multiplicities trivially distinguish graphs of different order, we only consider graphs of equal order ($|V(G)| = |V(H)|$). Without assumptions on node features or graph structures, two graphs in D could differ by a single node embedding at layer (j, i), which could be degraded by a single bit flip (Section 2.1).

To construct the graph-level upper bound (Theorem 2) from Theorem 1, note that two graphs of equal order are indistinguishable by the GNN only if they share identical unique node embeddings with corresponding multiplicities. The number of unique node embeddings a GNN assigns at layer (j, i) depends on both structure and node features. For MP-based GNNs, this can be bounded by the number of unique WL colors after j iterations, upon which a difference measure is defined using the symmetric difference of multisets.

Definition 2 (WL difference)**.** *For two labeled graphs G and H, we call $\Delta_{WL}(G, H, t) = |(C_l^{(t)}(G) \cup C_l^{(t)}(H)) \setminus (C_l^{(t)}(G) \cap C_l^{(t)}(H))|$ the* WL difference *at iteration t. Then, the two equal order graphs $G_e, H_e \in D$ with the highest WL difference of any two graphs in D at k iterations are given by $G_e, H_e = \arg\max_{G,H \in D, |V(G)|=|V(H)|} \Delta_{WL}(G, H, k)$.*

Theorem 2 (Graph-level upper bound)**.** *Let $\Phi^{(k)}(\mathbf{A}, \mathbf{X})\colon D \to Q^{m_k}$ be a maximally expressive GNN. Consider the graphs G_e and H_e defined as $G_e, H_e = \arg\max_{G,H \in D, |V(G)|=|V(H)|} \Delta_{WL}(G, H, j)$, where $e_j = \Delta_{WL}(G_e, H_e, j)$ represents the WL difference, and $d_{j,i} = \max_{\mathbf{x}_u, \mathbf{x}_v \in Q^{n_{j,i}}} \|\mathbf{x}_u - \mathbf{x}_v\|_0$ for $u \in V(G_e)$ and $v \in V(H_e)$. Then, the graph-level expressivity of $\Phi^{(k)}$ can be degraded by $\mathcal{O}(e_j \cdot d_{j,i} \cdot m_{j,i} \cdot b)$ bit flips, where b is the bit width.*

Similar to Theorem 1, Theorem 2 targets a sufficient condition for maximal expressivity defined in Definition 1. For a GNN with a ReadOut function only using the embeddings of the last GNN layer, which would differ from Equation (2) but be closest to the WL algorithm, Theorem 2 yields bounds for the guaranteed degradation of expressivity for bit flips in the

last layer. As for Theorem 1, we provide computational complexity estimates for Theorem 2 in the supplementary material.

2.3 Special Cases

The preceding analysis becomes more specific when considering assumptions about activation functions, dataset, and vulnerable layers, refining our understanding of the system's robustness and expressivity

ReLU Activations While PwL-activated MLPs can never induce a globally moment-injective function (Amir et al., 2024), they are highly popular. Specifically, ReLU-activated MLPs are subject to certain constraints on width and weights to ensure injectivity (Puthawala et al., 2022), even on constrained datasets, creating additional opportunities for an attacker to degrade GNN expressivity. Since width restrictions are more relevant in the global setting (Puthawala et al., 2022), which differs from our assumption of a bounded input domain, we focus on generally applicable weight limitations.

Lemma 3 (Equivalence of conditions for layer injectivity (ReLU))**.** *Let $\mathbf{W}^{(j,i)} \in U^{m_{j,i} \times n_{j,i}}$ be a linear transformation with $1 < n_{j,i} \leq m_{j,i}$, and let $R_{\mathbf{W}^{(j,i)}} = \{\mathbf{w}_r\}_{r=0}^{m_{j,i}}$ denote the set of row vectors of $\mathbf{W}^{(j,i)}$. Consider the ReLU activation function $\mathrm{ReLU} : U \to Q^+$. Then, the function $\mathrm{ReLU} \circ \mathbf{W}^{(j,i)} : Q^{n_{j,i}} \to Q^{m_{j,i}}$ is injective if and only if for any distinct $\mathbf{x}_u, \mathbf{x}_v \in Q^{n_{j,i}}$, there exists a row vector $\mathbf{w}_r \in R_{\mathbf{W}^{(j,i)}}$ such that $\langle \mathbf{x}_u, \mathbf{w}_r \rangle \neq \langle \mathbf{x}_v, \mathbf{w}_r \rangle$ and $\langle \mathbf{x}_u, \mathbf{w}_r \rangle > 0$ or $\langle \mathbf{x}_v, \mathbf{w}_r \rangle > 0$.*

The requirement for the dot product to be positive, as stated in Lemma 3, makes the GNN highly vulnerable. Specifically, in a signed representation, flipping the sign bits in the target weights will suffice, letting ReLU zero out the activations. This reduces the node-level upper bound (Theorem 1) from $\mathcal{O}(d_{j,i} \cdot m_{j,i} \cdot b)$ to $\mathcal{O}(d_{j,i} \cdot m_{j,i})$, independent of bit width b, as there is at most one sign bit per weight element. Similarly, it reduces the graph-level upper bound (Theorem 2) from $\mathcal{O}(e_j \cdot d_{j,i} \cdot m_{j,i} \cdot b)$ to $\mathcal{O}(e_j \cdot d_{j,i} \cdot m_{j,i})$.

Attacks on the First Layer A common assumption in literature is that node features are one-hot encodings of graph node labels (Xu et al., 2019), as seen in many real-world benchmark datasets (Morris et al., 2020; Hu et al., 2020). We examine how this encoding influences GNN expressivity vulnerability, particularly in the first layer. Assume the dataset has g unique node labels, so the dimension of the one-hot encoded vectors is g. Let $d = \max_{G \in D}(\deg_{\max}(G))$ denote the maximum node degree across all graphs in the dataset. The impact of these characteristics on the first layer of a GNN is captured in the following corollary.

Corollary 1 (Node-Level upper bound (first layer))**.** *Let $\Phi^{(k)}$ be a GNN and let the node features of all $G \in D$ be represented as one-hot encodings of the graph's node labels. Assume the dataset has g unique node labels, and let $d = \max_{G \in D}(\deg_{\max}(G))$ denote the maximum node degree across all graphs in the dataset. Then, the expressivity of the first layer $\mathrm{MLP}_l^{(1)}$ can be compromised by flipping $\mathcal{O}(m_{1,1} \cdot b \cdot nz)$ bits, where $nz = \min(2 \cdot d, n_{1,1})$, $m_{1,1}$ is the number of output dimensions, and b is the bit width.*

The required number of bit flips decreases as the difference between d and g increases, since $n_{1,1} = g$ is fixed for the first layer. This reflects a specialized case of the general node-level upper bound in Theorem 1, where vulnerability is heightened by the sparse, structured nature of one-hot encoded inputs. For the generalization from Theorem 1 to Theorem 2, note that Theorem 2 (like Theorem 1, without assumptions on node features or attacked layer) derives the number of elements to zero out per neuron as $d_{j,i} = \max_{\mathbf{x}_u, \mathbf{x}_v \in Q^{n_{j,i}}} |\mathbf{x}_u - \mathbf{x}_v|_0$, $u \in V(G_e), v \in V(H_e)$ with $G_e, H_e = \mathrm{argmax}_{G,H \in D, |V(G)|=|V(H)|} \Delta_{WL}(G, H, j)$. Assuming $\mathbf{W}^{(1,1)}$ is attacked and node features are one-hot encodings, we can bound $|\mathbf{x}_u - \mathbf{x}_v|_0$ by the same argument.

The Impact of Homophily To refine the node-level upper bound in Theorem 1, particularly for attacks on the input layer with one-hot encoded inputs, we incorporate the dataset's graph homophily ratio. The homophily ratio quantifies the likelihood that connected nodes share the same label (not to be confused with the class labels of nodes), measuring similarity within the graph structure. This relationship is formalized in the following corollary.

Corollary 2 (Node-Level upper bound (homophily))**.** *Let H_G denote the homophily ratio of a graph $G(V, E)$ with node labels l, defined as $H_G = |E(G)|^{-1} |\{(v_i, v_j) \in E(G) \mid l(v_i) = l(v_j)\}|$. For a dataset D, the average homophily ratio is $H_D = |D|^{-1} \sum_{G \in D} H_G$. Let P_D denote the probability that two randomly chosen nodes from $V_\cup(D)$ are connected, given by $P_D = \sum_{G \in D} |E(G)| \cdot \binom{|V(G)|}{2}^{-1}$. The number of bit flips required to make two nodes $u, v \in V_\cup(D)$ indistinguishable in the first layer is bounded by $\mathcal{O}(m_{1,1} \cdot b \cdot nz_H)$, where $nz_H = \min(2 \cdot d \cdot (1 - H_D) \cdot (1 - P_D), n_{1,1})$, d is the maximum node degree, $m_{1,1}$ is the output dimension of the first layer, and b is the bit width.*

The homophily ratio H_D influences the sparsity of the aggregates: a lower homophily ratio increases the likelihood that neighboring nodes have different labels, leading to fewer common non-zero entries in the embeddings of connected nodes. Since nz_H is typically less than the general case nz (unless $H_D = P_D = 1$,

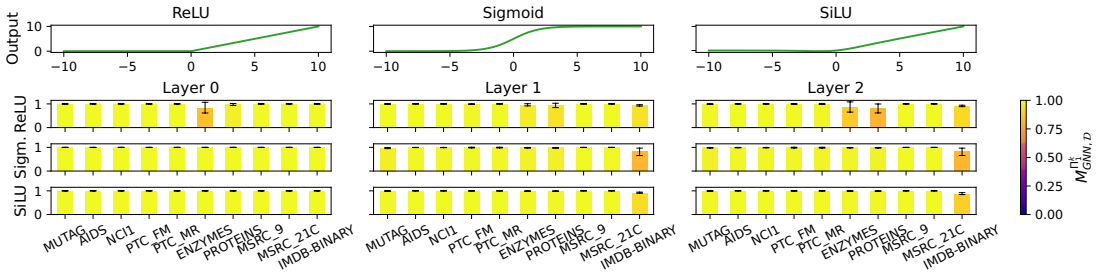


Figure 1: Visualization of $\mu \pm \sigma$ of the metric $M^{\Pi_1^k}_{GNN,D}$ (bottom) computed from the 20250 unperturbed (clean) models used in the attack runs for ReLU, Sigmoid, SiLU (top), aggregated across 3-layer GIN/GCN and DS.

which is rare), the refined bound $\mathcal{O}(m_{1,1} \cdot b \cdot nz_H)$ usually offers a tighter estimate for vulnerability when homophily is significant. This analysis shows how the dataset's structure and homophily can exacerbate GNN expressivity vulnerability, particularly with one-hot encoded inputs, refining the bounds in Theorem 1 and allowing extension to Theorem 2.

Note that the preceding formal analysis of the impact of homophily and attacks on the first layer might not extend to GAT, as the pre-transformation application of attention parameters and nonlinear transformations during aggregation might reduce or entirely negate these influence factors, and more analysis is needed to fully clarify their impact on these architectures.

3 EXPERIMENTS

We design our experiments to investigate key research questions (RQs) derived from our theoretical analysis, using 10 real-world datasets from TUDataset (Morris et al., 2020). These datasets, detailed in the supplementary material, are popular in contemporary research, feature one-hot or binary encoded node labels, and cover a range of common tasks in drug development (Xiong et al., 2021; Rossi et al., 2020), bioinformatics (Borgwardt et al., 2005), object recognition (Rossi and Ahmed, 2015), and social networks.

Specifically, we explore

(**RQ1**) whether the vulnerable bits in the floating-point representation depend on the activation function,

(**RQ2**) whether a loss of expressivity is linked to a loss of injectivity in the GNNs' MLPs, and

(**RQ3**) whether GNN resilience is connected to the graph partitioning induced by the WL coloring.

Moreover, we examine

(**RQ4**) whether resilience to first layer attacks relates to homophily and feature dimensionality.

We assess our RQs on GIN (2 hidden layers per MLP) due to its direct relation to our theory as well as on an architecture directly derived from DeepSets implementing Equation 1, which we abbreviate as DS (likewise 2 hidden layers per MLP). To demonstrate that (as claimed in Section 1) our theory extends to GCN, representing the class of spectral GNNs, we include GCN in our assessment. We use ReLU, Sigmoid and SiLU (Apicella et al., 2021) activations and initialize network parameters $\mathbf{W}^{(j,i)}$ randomly from a uniform distribution $\mathcal{U}(-\sqrt{\frac{1}{m_{j_i}}}, \sqrt{\frac{1}{m_{j_i}}})$ following common variance scaling initialization schemes (Glorot and Bengio, 2010; He et al., 2015). As GNNs are maximally expressive for randomly initialized weights in almost all cases (Amir et al., 2024) and our theoretical upper bounds, including those taking influence factors such as homophily into account, hold for all possible weights, we do not train our GNNs. Bits are flipped semi-randomly in either the sign, exponent, or the mantissa of the `FLOAT32` represented weights, to which we refer as target bits: specifically and in line with our theory (Section 1), we randomly induce $0 \to 1$ flips in sign bits and $1 \to 0$ flips in mantissa and exponent bits. For each experiment, we randomly initialize 5 GNN instances and repeat the induction of a certain percentage of bit flips in the target bits 5 times per instance. This results in 25 runs per experiment in total, reducing stochastic effects. Our experiments[1] took 8 weeks with three parallel workers to conclude and were conducted on a local server equipped with an NVIDIA H100 PCIe GPU (80GB VRAM), an Intel Xeon Gold 6326 CPU (500GB RAM) and a 1TB SSD.

Metrics We measure graph level expressivity (abbreviated *Exp*) as the percentage of non-isomorphic graphs of a dataset for which the GNN's final MP layer outputs node embeddings such that their sum is distinguishable for `FLOAT32` ($\epsilon_{mach} = 1.19 \times 10^{-7}$), which

[1]Our code is available in our repository at `https://github.com/lorenz0890/exproaistats`.

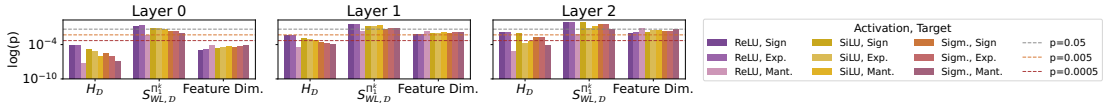


Figure 2: Significance levels of Spearman correlation of dataset properties $S^{(k)}_{WL,D}$, H_D and feature dimensionality with ΔExp computed across all runs and datasets for exponent, sign and mantissa target bits in Sigmoid, ReLU and SiLU activated 3-layer GIN/GCNs and DS (101250 runs in total).

necessitates that they have at least one distinguishable node embedding element. We do not introduce an additional tolerance level, as the impact of numerical error propagation on GNN expressivity has not yet been sufficiently investigated. This is equal to the assumption of a ReadOut function that only encodes the last layer's embeddings and enables us to pinpoint how bit flips in specific layers affect expressivity. We further measure the degree of refinement of the partitions induced by the GNN embeddings, as well as the 1-WL colors. Specifically, we use the *average WL subdivision ratio* $S^{(t)}_{WL,D} = |D|^{-1}\sum_{G\in D}\frac{|C^{(t)}_l(G)|}{|C^{(t-1)}_l(G)|}$, $t \geq 1$, and the *average GNN subdivision ratio* $S^{(k)}_{GNN,D} = |D|^{-1}\sum_{(\mathbf{A},\mathbf{X})\in D}\frac{|\Phi^{(k)}(\mathbf{A},\mathbf{X})|_\delta}{|\Phi^{(k-1)}(\mathbf{A},\mathbf{X})|_\delta}$, $k \geq 1$, of a given dataset D to measure 1-WL's and the GNN's embeddings partition diversity, respectively, whereby the $|\cdot|_\delta$ describes the number of elementwise distinguishable node embeddings vectors of a given graph up to ϵ_{mach}.

For specific k, it is possible that $S^{(k)}_{GNN,D} \geq S^{(k)}_{WL,D}$ since the GNN could recover from previous failures to distinguish certain unfolding trees if a layer's preceding functions were not injective (Kummer et al., 2024). However, it generally holds that $\prod_{k=1}^{n} S^{(k)}_{GNN,D} = S^{\Pi^n_1}_{GNN,D} \leq S^{\Pi^n_1}_{WL,D} = \prod_{k=1}^{n} S^{(k)}_{WL,D}$ since WL is the upper bound of MP GNN expressivity.

Since the injectivity of a function by itself can not be quantified, we introduce a substitute to determine the degree to which the $\text{MLP}^{(j)}_l$ of a GNN are capable to produce a 1-to-1 mapping of their input domain to their output domain. That is, to measure the mapping of functions learned by $\text{MLP}^{(j)}_l$'s at the jth layer, we measure the ratio of uniquely mapped inputs across the entire dataset D using the *unique mapping ratio* $M^{(j)} = |H^{(j)}|_\delta |Z^{(j)}|^{-1}_\delta$, which represents the ratio of inputs uniquely mapped (up to ϵ_{mach}) by the $\text{MLP}^{(j)}_l$, whereby $Z^{(j)}$ is the set of all inputs the $\text{MLP}^{(j)}_l$ receives on a dataset, representing node embeddings of graphs in D at layer $j-1$, and $H^{(j)}$ the set of all outputs $\text{MLP}^{(j)}_l$ produces for these inputs.

Clearly, $\text{MLP}^{(j)}_l$ is injective w.r.t. to its input domain if $M^{(j)} = 1$. We compute an aggregate score as $M^{\Pi^k_1}_{GNN,D} = \prod_{j=1}^{k} M^{(j)}$. Then, $M^{\Pi^k_1}_{GNN,D} = 1$ implies injectivity of all layers, hence Lemma 1's criteria are satisfied and consequently, GNN can be maximally expressive in the sense of Definition 1. We find $M^{\Pi^k_1}_{GNN,D}$ is nearly 1 for randomly initialized GNNs in most cases, see Figure 1, which is in line with the results of Amir et al. (2024). Note that $M^{\Pi^k_1}_{GNN,D} = 1$ implies $S^{\Pi^k_1}_{GNN}(D) = S^{\Pi^k_1}_{WL}(D)$ for maximally expressive MP based GNNs such as GIN (but not vice versa).

3.1 Results

Our empirical results (Figure 3) confirm our theoretical predictions.

(**RQ1**) ReLU-activated GNNs are particularly susceptible to bit flips, including sign bit flips, distinguishing them from Sigmoid- or SiLU-activated GNNs, as predicted by Lemmas 2 and 3. This vulnerability extends to upper layers, where expressivity degradation cannot be guaranteed for injective activations (Section 2.3), even for $M^{\Pi^k_1}_{GNN,D} < 1$. Attacks on the last layer's exponents degrade expressivity across all GNNs, datasets, and activations. Upper-layer vulnerability to bit flips in the exponent was only observed for non-injective activations ReLU and SiLU.

(**RQ2**) A relative decrease in injectivity is linked with expressivity degradation: with a Spearman correlation coefficient of 0.5209 across all experiments, there is a strong correlation between $\Delta M^{\Pi^k_1}_{GNN,D}$ and ΔExp, significant at $p = 1.2 \cdot 10^{-49}$ ($p < 0.005$). This aligns with the predicted potential loss of moment injectivity (Proposition 1). While Spearman correlation is robust to non-linear relationships, Pearson correlation showed similar significance only for first-layer bit flips, with no significant correlations in deeper layers, indicating a more complex relationship.

(**RQ3**) No general direct link between resilience and the granularity of the WL color partitioning (represented by $S^{\Pi^k_1}_{WL,D}$) of a dataset's graphs is established for all layers and activations but only specific combinations thereof (Figure 2). However, the reduction in GNN subdivision

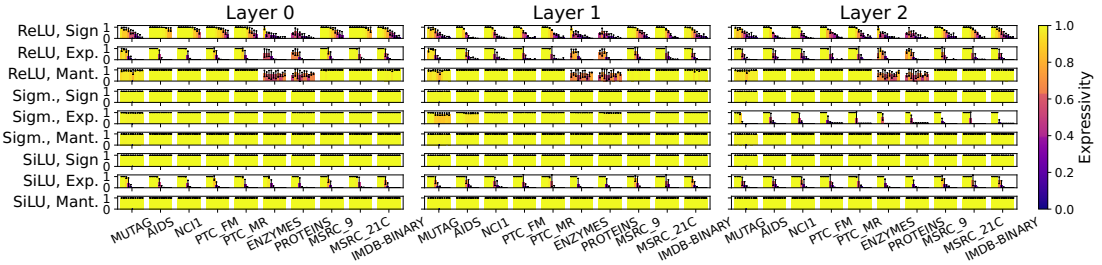


Figure 3: Flips in exponent, sign and mantissa bits of Sigmoid, ReLU and SiLU activated 3-layer GIN/GCNs and DS. Each group of 10 bars represents expressivity (i.e. the fraction of distinguishable non-isomorphic graphs of the given dataset) after (left to right) 1% to 95% bits flipped in a certain component of the `FLOAT32` representation of the weights in a layer. Each bar represents $\mu \pm \sigma$ computed from 25 runs (101250 runs in total).

ratio $\Delta S^{\Pi_1^k}_{GNN,D}$ is highly correlated with ΔExp, with a coefficient of 0.7390 at $p = 1.9 \cdot 10^{-319}$, indicating an indirect link to $S^{\Pi_1^k}_{GNN,D}$'s upper bound $S^{\Pi_1^k}_{WL,D}$ and suggesting that practical resilience is influenced by additional factors.

(**RQ4**) H_D and feature dimensionality show a low but highly significant ($p < 0.005$) correlation with ΔExp for first layer bit flips (Figure 2) as expected by Corollaries 1 and 2, with coefficients 0.099 and -0.088, respectively. This suggests that higher homophily correlates with increased expressivity loss, while increased feature dimensionality is protective. No significant correlation between H_D or feature dimensionality and ΔExp is found in deeper layers.

Implications for Practitioners Our experiments confirm the importance of homophily, feature dimensionality, and the GNN/1-WL subdivision ratio in influencing GNN resilience, indirectly validating the theoretical framework and suggesting that the broad upper bounds provided by our main Theorems 1 and 2 capture critical aspects relevant to real-world datasets. While expressivity loss may not always reduce practical performance, whenever bit-flip-induced expressivity loss does degrade predictive quality, our bounds on the required number of bit flips remain valid. In particular, any performance drop due to expressivity loss occurs with fewer bit flips than our worst-case estimates.

The observed correlations underscore the significance of these factors in designing robust GNNs, especially where expressivity is crucial. As our analysis applies broadly to any architecture relying on injective neural moments to distinguish multisets, our results offer actionable insights for practitioners: ReLU-activated GNNs' expressivity is more vulnerable to bit flips, particularly due to feature dimensionality and encoding schemes. To enhance robustness, practitioners could adopt SiLU activations as a resilient alternative to ReLU or increase feature dimensionality through precoloring techniques, such as (Alvarez-Gonzalez et al., 2022; Cai and Wang, 2019). Additionally, transforming one-hot encoded features into a dense format while preserving linear independence could maintain expressivity and improve robustness (see Section 2).

4 CONCLUSION

We establish theoretical foundations to understand the resilience of neural moment-based GNNs against BFAs and provide provable bounds for the number of bit flips required to degrade expressivity. We show that GNN robustness to BFAs is fundamentally influenced by architectural properties, the graph dataset's structural properties, and the activation functions employed. We find that ReLU-activated GNNs are particularly susceptible to BFAs, whereas employing alternative activation functions, such as SiLU or Sigmoid, enhances resilience. We further find that GNN resilience is significantly influenced by the dataset's homophily and feature dimensionality. Specifically, higher homophily tends to exacerbate the impact of bit flips, whereas higher feature dimensionality increases resilience. These insights underline the importance of considering both graph topology and feature space design when developing GNNs for applications where bit flip-robustness is paramount. Future work will explore the relationship between these theoretical factors and practical considerations, such as class label distributions, model training, optimization algorithms, learning rates, weight initialization, and regularization, deepening our understanding of practical vulnerabilities and their mitigation.

5 Acknowledgments

This work was supported by the Vienna Science and Technology Fund (WWTF) [10.47379/VRG19009].

References

Alvarez-Gonzalez, N., Kaltenbrunner, A., and Gómez, V. (2022). Beyond 1-WL with local ego-network encodings. In *The First Learning on Graphs Conference*.

Amir, T., Gortler, S., Avni, I., Ravina, R., and Dym, N. (2024). Neural injective functions for multisets, measures and graphs via a finite witness theorem. *Advances in Neural Information Processing Systems*, 36.

Apicella, A., Donnarumma, F., Isgrò, F., and Prevete, R. (2021). A survey on modern trainable activation functions. *Neural Networks*, 138:14–32.

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S., Smola, A. J., and Kriegel, H.-P. (2005). Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56.

Breier, J., Hou, X., Jap, D., Ma, L., Bhasin, S., and Liu, Y. (2018). Practical fault attack on deep neural networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, page 2204–2206. Association for Computing Machinery.

Cai, C. and Wang, Y. (2019). A simple yet effective baseline for non-attributed graph classification. *ICLR 2019 workshop on Representation learning*.

D'Inverno, G. A., Bianchini, M., Sampoli, M. L., and Scarselli, F. (2021). A unifying point of view on expressive power of gnns. *CoRR*, abs/2106.08992.

Gao, J., Lyu, T., Xiong, F., Wang, J., Ke, W., and Li, Z. (2022). Predicting the survival of cancer patients with multimodal graph neural network. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 19(2):699–709.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Hector, K., Moëllic, P.-A., Dumont, M., and Dutertre, J.-M. (2022). A closer look at evaluating the bit-flip attack against deep neural networks. In *2022 IEEE 28th International Symposium on On-Line Testing and Robust System Design*, pages 1–5.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. (2020). Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133.

Jegelka, S. (2022). Theory of graph neural networks: Representation and learning. *CoRR*, abs/2204.07697.

Jiao, X., Wang, R., Lin, F., Moore, D., and Sankar, S. (2022). Assessing and analyzing the resilience of graph neural networks against hardware faults. *CoRR*, abs/2212.03475.

Jin, W., Li, Y., Xu, H., Wang, Y., Ji, S., Aggarwal, C., and Tang, J. (2021). Adversarial attacks and defenses on graphs. *ACM SIGKDD Explorations Newsletter*, 22(2):19–34.

Khare, Y., Lakara, K., Inukonda, M. S., Mittal, S., Chandra, M., and Kaushik, A. (2022). Design and analysis of novel bit-flip attacks and defense strategies for dnns. In *2022 IEEE Conference on Dependable and Secure Computing*, pages 1–8.

Kummer, L., Moustafa, S., Schrittwieser, S., Gansterer, W., and Kriege, N. (2024). Attacking graph neural networks with bit flips: Weisfeiler and Leman go indifferent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1428–1439.

Lu, H. and Uddin, S. (2021). A weighted patient network-based framework for predicting chronic diseases using graph neural networks. *Scientific reports*, 11(1):22607.

Ma, J., Ding, S., and Mei, Q. (2020). Towards more practical adversarial attacks on graph neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 4756–4766. Curran Associates, Inc.

Morris, C., Fey, M., and Kriege, N. (2021). The power of the Weisfeiler-Leman algorithm for machine learning with graphs. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4543–4550. International Joint Conferences on Artificial Intelligence Organization.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. (2020). TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*.

Morris, C., Lipman, Y., Maron, H., Rieck, B., Kriege, N. M., Grohe, M., Fey, M., and Borgwardt, K. (2023). Weisfeiler and Leman go machine learning: The story so far. *Journal of Machine Learning Research*, 24(333):1–59.

Mutlu, O. and Kim, J. S. (2019). Rowhammer: A retrospective. *IEEE Transactions on Computer-Aided De-*

sign of Integrated Circuits and Systems, 39(8):1555–1571.

Neumann, M., Garnett, R., Bauckhage, C., and Kersting, K. (2016). Propagation kernels: efficient graph kernels from propagated information. *Machine learning*, 102:209–245.

Puthawala, M., Kothari, K., Lassas, M., Dokmanić, I., and De Hoop, M. (2022). Globally injective relu networks. *The Journal of Machine Learning Research*, 23(1):4544–4598.

Qian, C., Zhang, M., Nie, Y., Lu, S., and Cao, H. (2023). A survey of bit-flip attacks on deep neural network and corresponding defense methods. *Electronics*, 12(4):853.

Rakin, A. S., He, Z., and Fan, D. (2019). Bit-flip attack: Crushing neural network with progressive bit search. In *2019 IEEE/CVF International Conference on Computer Vision*, pages 1211–1220.

Rossi, R. A. and Ahmed, N. K. (2015). The network data repository with interactive graph analytics and visualization. In *AAAI*.

Rossi, R. A., Jin, D., Kim, S., Ahmed, N. K., Koutra, D., and Lee, J. B. (2020). On proximity and structural role-based embeddings in networks: Misconceptions, techniques, and applications. *ACM Transactions on Knowledge Discovery from Data*, 14(5):1–37.

Roth, A. and Liebig, T. (2024). Rank collapse causes over-smoothing and over-correlation in graph neural networks. In Villar, S. and Chamberlain, B., editors, *Proceedings of the Second Learning on Graphs Conference*, volume 231 of *Proceedings of Machine Learning Research*, pages 35:1–35:23. PMLR.

Schomburg, I., Chang, A., and Schomburg, D. (2002). Brenda, enzyme data and metabolic information. *Nucleic acids research*, 30(1):47–49.

Schulz, T. H., Horváth, T., Welke, P., and Wrobel, S. (2022). A generalized Weisfeiler-Lehman graph kernel. *Machine Learning*, 111(7):2601–2629.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. (2011). Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research*, 12(9).

Sun, Y., Wang, S., Tang, X., Hsieh, T.-Y., and Honavar, V. (2020). Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. page 673–683.

Sun, Z., Yin, H., Chen, H., Chen, T., Cui, L., and Yang, F. (2021). Disease prediction via graph neural networks. *IEEE Journal of Biomedical and Health Informatics*, 25(3):818–826.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations*.

Wagstaff, E., Fuchs, F. B., Engelcke, M., Osborne, M. A., and Posner, I. (2022). Universal approximation of functions on sets. *J. Mach. Learn. Res.*, 23:151:1–151:56.

Welling, M. and Kipf, T. N. (2016). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Wu, B., Yuan, X., Wang, S., Li, Q., Xue, M., and Pan, S. (2023). Securing graph neural networks in mlaas: A comprehensive realization of query-based integrity verification. *arXiv preprint arXiv:2312.07870*.

Wu, L., Cui, P., Pei, J., Zhao, L., and Song, L. (2022). *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. (2018). Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530.

Xiong, J., Xiong, Z., Chen, K., Jiang, H., and Zheng, M. (2021). Graph neural networks for automated de novo drug design. *Drug Discovery Today*, 26(6):1382–1393.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). How powerful are graph neural networks? In *7th International Conference on Learning Representations*.

Yanardag, P. and Vishwanathan, S. (2015). Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1365–1374.

Yao, F., Rakin, A. S., and Fan, D. (2020). Deephammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips. In *USENIX Security Symposium*.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Póczos, B., Salakhutdinov, R., and Smola, A. J. (2017). Deep sets. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 3391–3401.

Zopf, M. (2022). 1-wl expressiveness is (almost) all you need. In *International Joint Conference on Neural Networks, IJCNN*, pages 1–8. IEEE.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes, see Sections 1 and 2.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Yes. Whereas we consider our two main theorems to be theoretical quantities, we provide time complexity for their concrete computation in the supplementary material.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. Yes, link on page 7 as footnote 1.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Yes, either included in the claim itself or the surrounding text.
   (b) Complete proofs of all theoretical results. Yes, included in the supplementary material.
   (c) Clear explanations of any assumptions. Yes, all assumptions are clearly specified in the textual description of each theoretical claim or the claim itself.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). Yes, link on page 7 as footnote 1.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Yes, see Sec 3.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes, see Sec 3.
   (d) A description of the computing infrastructure used. Yes, see Sec 3.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. Yes, see Sec 3.
   (b) The license information of the assets, if applicable. Yes, licenses are included in the libraries we used.
   (c) New assets either in the supplemental material or as a URL, if applicable. Yes, link to code on page 7 as footnote 1.
   (d) Information about consent from data providers/curators. Not Applicable.
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Not Applicable.

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. Not Applicable.
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. Not Applicable.
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable.

A PROOFS

Proof of Proposition 1 We prove by contradiction that $(\forall A \in \Omega_n^d | \sum_{\mathbf{x} \in S(A)} f(\mathbf{x}) \lambda_{\mathbf{x}} = \mathbf{0} \Rightarrow \forall \mathbf{x} \in A | \lambda_{\mathbf{x}} = 0, \lambda_{\mathbf{x}} \in V) \implies (\forall A, B \in \Omega_n^d | \hat{f}(A) = \hat{f}(B) \Rightarrow A = B$, $n \geq 2)$

Assume the negation of the necessary condition:
(1.) $\exists A, B \in \Omega_n^d | A \neq B \wedge \hat{f}(A) = \hat{f}(B)$.

Then, through transformations, we reach:
(2.) $\hat{f}(A) = \hat{f}(B) \Leftrightarrow$
(3.) $\sum_{\mathbf{x} \in A} f(\mathbf{x}) = \sum_{\mathbf{x} \in B} f(\mathbf{x}) \Leftrightarrow$
(4.) $\sum_{\mathbf{x} \in S(A)} m_A(\mathbf{x}) f(\mathbf{x}) - \sum_{\mathbf{x} \in S(B)} m_B(\mathbf{x}) f(\mathbf{x}) = \mathbf{0} \Leftrightarrow$
(5.) $\sum_{\mathbf{x} \in S_\cup} m_A(\mathbf{x}) f(\mathbf{x}) - \sum_{\mathbf{x} \in S_\cup} m_B(\mathbf{x}) f(\mathbf{x}) = \mathbf{0} \Leftrightarrow$
(6.) $\sum_{\mathbf{x} \in S_\cup} (m_A(\mathbf{x}) - m_B(\mathbf{x})) f(\mathbf{x}) = \mathbf{0}$
with $S_\cup = S(A \cup B)$ but without guarantee for $S_\cup \in \Omega_n^d$ since $|S_\cup| \leq 2n$.

However, because $A \neq B \Rightarrow |A - B| > 0$, it follows that
(7.) $\exists \mathbf{x} \in S_\cup | (m_A(\mathbf{x}) - m_B(\mathbf{x})) \neq 0$ and consequentially, to satisfy (6.), that
(8.) $\exists \mathbf{y} \in S_\cup, \mathbf{y} \neq \mathbf{x}, (m_A(\mathbf{y}) - m_B(\mathbf{y})) \neq 0 | (m_A(\mathbf{x}) - m_B(\mathbf{x})) f(\mathbf{x}) + (m_A(\mathbf{y}) - m_B(\mathbf{y})) f(\mathbf{y}) = \mathbf{0}$.

Note we assume $f(\mathbf{y}) \neq f(\mathbf{x})$ in (8.) because $f(\mathbf{y}) = f(\mathbf{x})$ for $\mathbf{x} \neq \mathbf{y}$, would directly contradict $\forall A \in \Omega_n^d | \sum_{\mathbf{x} \in S(A)} f(\mathbf{x}) \lambda_{\mathbf{x}} = \mathbf{0} \Rightarrow \forall \mathbf{x} \in A | \lambda_{\mathbf{x}} = 0, \lambda_{\mathbf{x}} \in V$.

To proceed, choosing $M = \{\!\{\mathbf{y}, \mathbf{x}\}\!\} \subseteq S_\cup$, $|M| = 2 \leq n$ and thus $M \in \Omega_n^d$, leads to
(9.) $f(\mathbf{x}) \lambda_{\mathbf{x}} + f(\mathbf{y}) \lambda_{\mathbf{y}} = \mathbf{0}$ with $\lambda_{\mathbf{x}} = (m_A(\mathbf{x}) - m_B(\mathbf{x})) \neq 0$ and $\lambda_{\mathbf{y}} = -(m_A(\mathbf{y}) - m_B(\mathbf{y})) \neq 0$, $\mathbf{x}, \mathbf{y} \in M$, contradicting the sufficient condition that $\forall A \in \Omega_n^d | \sum_{\mathbf{x} \in S(A)} f(\mathbf{x}) \lambda_{\mathbf{x}} = \mathbf{0} \Rightarrow \forall \mathbf{x} \in A | \lambda_{\mathbf{x}} = 0$.

□

Proof of Lemma 2 If for arbitrary $\mathbf{x}_u, \mathbf{x}_v \in Q^{n_{j,i}}, \mathbf{x}_u \neq \mathbf{x}_v$ it holds that $\exists \mathbf{w}_r \in R_{\mathbf{W}^{(j,i)}}$ s.t. $\langle \mathbf{x}_u, \mathbf{w}_r \rangle \neq \langle \mathbf{x}_v, \mathbf{w}_r \rangle$, then $\sigma(\mathbf{W}^{(j,i)} \mathbf{x}_u) \neq \sigma(\mathbf{W}^{(j,i)} \mathbf{x}_v)$ follows directly from σ's injectivity. Conversely, assume $\forall \mathbf{x}_u, \mathbf{x}_v \in Q^{n_{j,i}}$ with $\mathbf{x}_u \neq \mathbf{x}_v$ it holds that $\sigma(\mathbf{W}^{(j,i)} \mathbf{x}_u) \neq \sigma(\mathbf{W}^{(j,i)} \mathbf{x}_v)$, then $\exists \mathbf{w}_r \in R_{\mathbf{W}^{(j,i)}}$ s.t. $\sigma(\langle \mathbf{x}_u, \mathbf{w}_r \rangle) \neq \sigma(\langle \mathbf{x}_v, \mathbf{w}_r \rangle)$. Because of σ's injectivity, $\neg(\sigma(\langle \mathbf{x}_u, \mathbf{w}_r \rangle) \neq \sigma(\langle \mathbf{x}_v, \mathbf{w}_r \rangle))$ would imply $\langle \mathbf{x}_u, \mathbf{w}_r \rangle = \langle \mathbf{x}_v, \mathbf{w}_r \rangle$ and further $\mathbf{x}_u = \mathbf{x}_v$. This would stand in contradiction to the assumption $\mathbf{x}_u \neq \mathbf{x}_v$, such that $\exists \mathbf{w}_r \in R_{\mathbf{W}^{(j,i)}}$ for which $\langle \mathbf{x}_u, \mathbf{w}_r \rangle \neq \langle \mathbf{x}_v, \mathbf{w}_r \rangle$ follows. □

Proof of Theorem 1 Let $\Phi^{(k)}$ be a maximally expressive GNN (Definition 1) with k layers as in (1). That is, each layer $\sigma \circ \mathbf{W}^{(j,i)}$ of each $\text{MLP}_l^{(j)}$ of $\Phi^{(k)}$ satisfies the conditions of Lemma 2. Let (j, i) be an arbitrary layer. Let $\mathbf{x}_u, \mathbf{x}_v \in Q^{n_{j,i}}, \mathbf{x}_u \neq \mathbf{x}_v$ be chosen such that the number of indices where $\mathbf{x}_u, \mathbf{x}_v$ differ elementwise $d_{j,i} = \max_{\mathbf{x}_u, \mathbf{x}_v \in Q^{n_{j,i}}} \|\mathbf{x}_u - \mathbf{x}_v\|_0$ is maximal for $u, v \in V_\cup(D)$. Then, at most $d_{j,i}$ weight elements of $\mathbf{W}^{(j,i)}$ with b bits each would have to be modified. In the worst case, that is setting the $d_{j,i}$ weights elements associated with the d indices where $\mathbf{x}_u, \mathbf{x}_v$ differ to 0 at the cost of at most b bit flips in each of the $m_{j,i}$ rows $\mathbf{w}_r$, amounting to at most $d_{j,i} \cdot m_{j,i} \cdot b$ bit flips after which $\Phi^{(k)}$'s maximal node-level expressivity can not be guaranteed anymore. □

Proof of Theorem 2 Let $G_e, H_e = \arg\max_{G,H \in D} \Delta_{WL}(G, H, j)$ and $e_j = \Delta_{WL}(G_e, H_e, j)$. That is, G_e and H_e are chosen from D such that they are structurally most differentiable by WL after j iterations, $e_j > 0 \Rightarrow G_e \not\simeq H_e$. Thus, also the outputs of the $\text{MLP}_l^{(j)}$, $j \leq k$, of a GNN $\Phi^{(k)}$ satisfying Definition 1 will differ in at most e_j embedding vectors. Then, as follows from Theorem 1, a given layer (j, i) will require at most $d_{j,i} \cdot m_{j,i} \cdot b$ bit flips with $d_{j,i} = \max_{\mathbf{x}_u, \mathbf{x}_v \in Q^{n_{j,i}}} \|\mathbf{x}_u - \mathbf{x}_v\|_0$, $u \in V(G_e), v \in V(H_e)$, to make a single pair of the e_j WL distinguishable nodes of G_e and H_e receive the same embeddings. If we repeat this step until it is guaranteed that none of the nodes of G_e and H_e can be distinguished pairwise at layer (j, i) of $\Phi^{(k)}$, we require at most $e_j \cdot d_{j,i} \cdot m_{j,i} \cdot b$ bit flips. □

Proof of Lemma 3 If for arbitrary $\mathbf{x}_u, \mathbf{x}_v \in Q^{n_{j,i}}, \mathbf{x}_u \neq \mathbf{x}_v$ it holds that $\exists \mathbf{w}_r \in R_{\mathbf{W}^{(j,i)}}$ s.t. $\langle \mathbf{x}_u, \mathbf{w}_r \rangle \neq \langle \mathbf{x}_v, \mathbf{w}_r \rangle$ and $\langle \mathbf{x}_u, \mathbf{w}_r \rangle \geq 0, \langle \mathbf{x}_v, \mathbf{w}_r \rangle \geq 0$, then $\text{ReLU}(\mathbf{W}^{(j,i)} \mathbf{x}_u) \neq \text{ReLU}(\mathbf{W}^{(j,i)} \mathbf{x}_v)$ follows directly from ReLUs acting as identity function on positive inputs. Conversely, assume $\forall \mathbf{x}_u, \mathbf{x}_v \in Q^{n_{j,i}}$ with $\mathbf{x}_u \neq \mathbf{x}_v$ it holds that $\text{ReLU}(\mathbf{W}^{(j,i)} \mathbf{x}_u) \neq \text{ReLU}(\mathbf{W}^{(j,i)} \mathbf{x}_v)$, then $\exists \mathbf{w}_r \in R_{\mathbf{W}^{(j,i)}}$ s.t. $\text{ReLU}(\langle \mathbf{x}_u, \mathbf{w}_r \rangle) \neq \text{ReLU}(\langle \mathbf{x}_v, \mathbf{w}_r \rangle)$. Because ReLU acts as identity for positive inputs and zeros out negative inputs, it follows that $\langle \mathbf{x}_u, \mathbf{w}_r \rangle > 0 \vee \langle \mathbf{x}_v, \mathbf{w}_r \rangle > 0$ and $\langle \mathbf{x}_u, \mathbf{w}_r \rangle \neq \langle \mathbf{x}_v, \mathbf{w}_r \rangle$. □

Proof of Corollary 1 Assume that the dimension of the one-hot encoded node label vectors is g. After sum aggregation in the first layer's $\text{MLP}_l^{(1)}$, the number of non-zero elements (i.e., the number of distinct contributions from neighboring nodes) for any given node is at most $\min(d, n_{1,1})$, where $n_{1,1}$ is the input dimension to the first layer. Consequently, the number of zero elements in the aggregation is at least $\max(g - d, 0)$.

Consider two nodes $u, v \in V_\cup(D)$ that have the largest number of common non-zero embedding entries, denoted as $nz = \min(2 \cdot d, n_{1,1})$. To make these nodes

indistinguishable after the application of $\text{MLP}_l^{(1)}$, it follows from Theorem 1 that $\mathcal{O}(m_{1,1} \cdot b \cdot nz)$ bit flips would be required in the first layer's MLP. □

Proof of Corollary 2 For a graph $G(V, E)$ with node labels l, the homophily ratio H_G quantifies the probability that two connected nodes share the same label. This ratio is averaged over a dataset D to yield the average homophily ratio H_D, which indicates the expected probability that any two connected nodes in $V_\cup(D)$ share the same label.

Next, we consider the probability P_D, representing the likelihood that two randomly chosen nodes from $V_\cup(D)$ are connected. For one-hot encoded input vectors, both the connectivity and homophily influence the non-zero entries in the node embeddings. Specifically, for any pair of nodes $u, v \in V_\cup(D)$, the number of non-zero entries in their embeddings can be estimated as $nz_H = \min(2 \cdot d \cdot (1 - H_D) \cdot (1 - P_D), n_{1,1})$, where d is the maximum node degree, and $n_{1,1}$ is the input dimension to the first layer. Consequently, the number of bit flips required to make these nodes indistinguishable in the first layer decreases. Thus, this leads to a more vulnerable GNN expressivity, specifically bounded by $\mathcal{O}(m_{1,1} \cdot b \cdot nz_H)$ as follows Theorem 1. □

B COMPUTATIONAL COMPLEXITY

While both, Theorem 1 and Theorem 2 provide theoretical upper bounds to GNN resilience and the main parameters influencing their computational complexity $d_{j,i}$ and e_j are hence primarily intended as analytical constructs, we provide estimates of the computational complexities involved in practically computing $d_{i,j}$ and e_j below.

To determine time complexity of calculating $d_{j,i}$ as used in Theorem 1, we analyze the expression $d_{j,i} = \max_{\mathbf{x}_u, \mathbf{x}_v \in Q^{n_{j,i}}} |\mathbf{x}_u - \mathbf{x}_v|_0$. This represents the maximum ℓ_0 norm of the difference between node embeddings within the space $Q^{n_{j,i}}$. The ℓ_0 norm measures the number of entries where two vectors $\mathbf{x}_u$ and $\mathbf{x}_v$ differ. The search space consists of all possible pairs of vectors in $Q^{n_{j,i}}$, yielding $|Q|^{2n_{j,i}}$ pairs. Calculating the ℓ_0 norm for each pair requires iterating over all $n_{j,i}$ dimensions, resulting in a time complexity of $\mathcal{O}(n_{j,i})$ for each pair. Consequently, the overall complexity for evaluating all pairs to find the maximum ℓ_0 norm is $\mathcal{O}(|Q|^{2n_{j,i}} \cdot n_{j,i})$. This reflects the worst-case scenario where every pair of node embeddings is considered. Thus, $d_{j,i}$ can be calculated in $\mathcal{O}(|Q|^{2n_{j,i}} \cdot n_{j,i})$ time.

To determine the time complexity of calculating both $d_{j,i}$ and e_j as used in Theorem 2, we must consider the process of identifying the graph pair G_e and H_e, which maximize the WL difference Δ_{WL}. The value $d_{j,i}$ is defined as the maximum ℓ_0 norm between node embeddings $\mathbf{x}_u$ and $\mathbf{x}_v$ from graphs G_e and H_e respectively, specifically $d_{j,i} = \max_{\mathbf{x}_u, \mathbf{x}_v \in Q^{n_{j,i}}} |\mathbf{x}_u - \mathbf{x}_v|_0$. The process starts by identifying G_e and H_e as the pair of graphs in the dataset D with equal numbers of vertices that maximize $\Delta_{WL}(G, H, j)$. This involves evaluating all graph pairs, leading to $\mathcal{O}(|D|^2)$ combinations, where $|D|$ is the number of graphs. For each pair, the WL difference calculation requires running the WL algorithm over k iterations with a complexity of $\mathcal{O}(k \cdot (n + m))$, where n is the number of vertices and m is the number of edges. Once G_e and H_e are identified, the complexity of calculating $d_{j,i}$ involves computing the ℓ_0 norm for each node pair across the two graphs, yielding a complexity of $\mathcal{O}(|V(G_e)|^2 \cdot n_{j,i})$. Thus, the overall complexity for $d_{j,i}$ is $\mathcal{O}(|D|^2 \cdot k \cdot (n_{max(D)} + m_{max(D)}) + |V(G_e)|^2 \cdot n_{j,i})$ with $n_{max(D)}$ is the maximal number of vertices and $m_{max(D)}$ is the maximal number of edges of any $G \in D$. For e_j, the complexity is primarily driven by identifying the pair G_e and H_e, resulting in $\mathcal{O}(|D|^2 \cdot k \cdot (n_{max(D)} + m_{max(D)}))$, capturing the need to evaluate the WL difference across all graph pairs. These complexities reflect the calculations needed for maximizing graph differentiation and node embedding discrepancies.

C RESILIENCE TO RANDOM BIT FLIPS

Above formulations on the bounds of the numbers of flips required to degrade graph level and node level expressivity allow us to formulate bounds on the probabilities P_{node} and P_{graph} that a random bit flip will degrade a GNNs node or graph expressivity. In the node level case, where node level expressivity would diminish guaranteed in $\mathcal{O}(\max_{\mathbf{x}_u, \mathbf{x}_v \in Q^{n_{k,l}}} \|\mathbf{x}_u - \mathbf{x}_v\|_0 \cdot m_{k,l} \cdot b)$ bit flips for $u, v \in V_\cup(D)$ by a dedicated attacker, for random flips we would have to replace b with 2^b, covering all possible combinations of flips in the (k, l)th layer of the GNN. Note we chose the (k, l)th layer here as only in this case a loss of injectivity would guarantee a loss of expressivity.

$$P_{node} \leq (n_{k,l} \cdot m_{k,l})^{-\beta_{node}} \tag{3}$$

with $\beta_{node} = \max_{\mathbf{x}_u, \mathbf{x}_v \in Q^{n_{k,l}}} \|\mathbf{x}_u - \mathbf{x}_v\|_0 \cdot m_{k,l} \cdot 2^b$ for $u, v \in V_\cup(D)$. Likewise, we can arrive at a similar formulation for graph level expressivity

$$P_{graph} \leq (n_{k,l} \cdot m_{k,l})^{-\beta_{graph}} \tag{4}$$

with $\beta_{graph} = e_k \cdot d_{k,l} \cdot m_{k,l} \cdot 2^b$ and $e_k = \Delta_{WL}(G_e, H_e, k)$, $d_{k,l} = \max_{\mathbf{x}_u, \mathbf{x}_v \in Q^{n_{j,i}}} \|\mathbf{x}_u - \mathbf{x}_v\|_0$, $u \in V(G_e), v \in V(H_e)$ for $G_e, H_e =$

Table 1: Overview of Selected Datasets

DATASET	TYPE	#GRAPHS	AVG. NODES	AVG. EDGES	LABELS	TASK
MUTAG	Chemical	188	17.9	19.8	Node, Edge	Class.
AIDS	Chemical	2,000	15.7	16.2	Node, Edge	Class.
PTC_FM	Chemical	349	14.1	14.5	Node, Edge	Class.
PTC_MR	Chemical	344	14.3	14.7	Node, Edge	Class.
NCI1	Chemical	4,110	29.9	32.3	Node	Class.
PROTEINS	Protein	1,113	39.1	72.8	Node	Class.
ENZYMES	Protein	600	32.6	62.1	Node	Class.
MSRC_9	Image	221	40.6	72.4	Node	Class.
MSRC_21C	Image	563	77.5	142.8	Node	Class.
IMDB-BINARY	Social	1,000	19.8	96.5	-	Class.

$\arg\max_{G,H\in D} \Delta_{WL}(G,H,k)$. Both equations (3) and (4), can be adjusted to the special cases of ReLU activations, homophily and one hot encoded inputs and general layers (i.e. not only the last layer), leading to lower bounds for the given probabilities.

D DATASETS

We use a diverse set of ten real-world datasets to evaluate the intricate interplay between robustness and expressivity of GNNs, with each dataset chosen for its relevance to specific domains and graph structures. These datasets are part of the well-established TUDataset collection (Morris et al., 2020), which has become a benchmark standard for graph classification and regression tasks. An overview of the datasets is provided in Table 1.

Chemical Compounds: The MUTAG, AIDS, PTC_FM, PTC_MR, and NCI1 datasets represent chemical compounds, where graphs model molecules with nodes as atoms and edges as chemical bonds. MUTAG is one of the earliest and most commonly used datasets for graph classification, consisting of nitroaromatic compounds classified based on their mutagenicity on Salmonella typhimurium. Similarly, the AIDS dataset comprises molecular graphs used in anti-HIV drug discovery, with the task of predicting inhibitory activity against HIV. The PTC datasets (FM and MR) include compounds tested for carcinogenicity on rodents, classified by different experimental protocols. NCI1 is a larger dataset derived from the National Cancer Institute's screening, where the task is to classify compounds based on their activity against non-small cell lung cancer.

Protein Structures: The PROTEINS and ENZYMES datasets originate from the bioinformatics domain, where the graphs represent protein structures. In PROTEINS, nodes represent secondary structure elements of proteins, such as helices and sheets, connected based on spatial proximity. The classification task involves predicting whether a protein is an enzyme or not. ENZYMES extends this concept by assigning enzymes to one of the six top-level classes of the Enzyme Commission (EC) based on the chemical reactions they catalyze (Borgwardt et al., 2005; Schomburg et al., 2002).

Image Segmentation: The MSRC_9 and MSRC_21C datasets are derived from the MSRC database and are used for semantic image segmentation tasks. Here, graphs are constructed from images where nodes represent superpixels and edges encode the spatial relationships between these superpixels. These datasets challenge GNNs to classify nodes into different categories based on visual content, with MSRC_21C being an extended version containing more classes and a higher complexity (Neumann et al., 2016).

Social Networks: The IMDB-BINARY dataset is a social network dataset where each graph represents the collaboration network of actors who have appeared together in movies. The task is to classify these networks based on the genre of the movies, specifically distinguishing between action and romance genres. This dataset exemplifies the challenges posed by real-world social networks, where nodes represent individuals, and edges signify their interactions, requiring the GNN to capture intricate social dynamics (Yanardag and Vishwanathan, 2015).

These datasets collectively provide a comprehensive evaluation platform, covering a range of different graph structures and complexities, from small molecular graphs to larger social networks and image-derived graphs. By utilizing such a diverse set of benchmarks, our empirical analysis robustly tests our hypothesis across multiple domains.

A.4 Weisfeiler and Leman Go Gambling

Title	Weisfeiler and Leman Go Gambling: Why Expressive Lottery Tickets Win
Authors	Lorenz Kummer, Samir Moustafa, Anatol Ehrlich, Franka Bause, Nikolaus Suess, Wilfried Gansterer, Nils Kriege
Venue	Proceedings of The 42nd International Conference on Machine Learning (CORE ranking: A*)
Identifier	ISSN:2640-3498
Division of work	Lorenz Kummer: Conceived the initial idea; formulated the problem and most theoretical derivations; designed and implemented experiments; conducted the evaluation; wrote the article and created most visualizations; produced the conference poster; reviewed the related work. Samir Moustafa: Participated in key theoretical derivations and key discussions on the paper's contribution on sparse training dynamics; participated in discussions and reviewed and refined the article and conference poster. Anatol Ehrlich: Provided domain expertise on the chemically sensible interpretation of the theoretical and empirical results for molecular graphs; contributed to related visualizations; participated in discussions; reviewed and refined the article and conference poster. Franka Bause: Reviewed and refined the article and conference poster. Nikolaus Suess: Designed and executed experiments; reviewed and refined the article and conference poster. Wilfried Gansterer: Provided mentoring and supervision. Nils Kriege: Participated in discussions; reviewed and refined the article and conference poster; provided mentoring and supervision.

Weisfeiler and Leman Go Gambling: Why Expressive Lottery Tickets Win

Lorenz Kummer [1 2] **Samir Moustafa** [1 2] **Anatol Ehrlich** [1 2] **Franka Bause** [1 2]
Nikolaus Suess [1] **Wilfried N. Gansterer** [1] **Nils M. Kriege** [1 3]

Abstract

The lottery ticket hypothesis (LTH) is well-studied for convolutional neural networks but has been validated only empirically for graph neural networks (GNNs), for which theoretical findings are largely lacking. In this paper, we identify the expressivity of sparse subnetworks, i.e. their ability to distinguish non-isomorphic graphs, as crucial for finding winning tickets that preserve the predictive performance. We establish conditions under which the expressivity of a sparsely initialized GNN matches that of the full network, particularly when compared to the Weisfeiler-Leman test, and in that context put forward and prove a *Strong Expressive Lottery Ticket Hypothesis*. We subsequently show that an increased expressivity in the initialization potentially accelerates model convergence and improves generalization. Our findings establish novel theoretical foundations for both LTH and GNN research, highlighting the importance of maintaining expressivity in sparsely initialized GNNs. We illustrate our results using examples from drug discovery.

1. Introduction

Graph Neural Networks (GNNs) have emerged as powerful tools for learning over graph-structured data. Complex objects such as molecules, proteins or social networks are represented as graphs. Nodes represent parts and edges their relations, both can be enriched with features. GNNs generalize established deep learning techniques and extend their applicability to new domains such as financial and social network analysis, medical data analysis, or chem- and bioinformatics (Lu & Uddin, 2021; Cheung & Moura, 2020; Sun et al., 2021; Gao et al., 2022; Wu et al., 2018; Xiong et al., 2021). One of the key fields in GNN research is expressivity, that is, a GNN's ability to distinguish non-isomorphic graphs. This expressivity is often benchmarked using the Weisfeiler-Leman (WL) test, with the 1-WL test being a standard baseline for many models. Key contributions to this field include Deep Sets (Zaheer et al., 2017), which introduced permutation-invariant learning on sets, and the Graph Isomorphism Network (GIN) (Xu et al., 2019), which explicitly aimed to match the expressivity of the 1-WL test. GIN set a new standard in graph representation learning by using a simple yet powerful aggregation function that ensures the model's ability to distinguish non-isomorphic graphs at a level comparable to the 1-WL test. The combination of these advances has shaped the current understanding of GNN expressivity, where models are judged by their capacity to approximate or exceed the performance of 1-WL in distinguishing complex graph structures. A large amount of research has been devoted to improving the expressivity of GNNs, aiming to go beyond the limitations of 1-WL and increase the model's capacity to distinguish between structurally distinct graphs (Morris et al., 2023). Despite these efforts, the 1-WL test continues to distinguish most graphs in widely used benchmarks (Morris et al., 2021; Zopf, 2022), demonstrating that, in many scenarios, it is not necessary to go beyond this foundational approach.

Practically, in drug discovery and molecular property prediction, the impact of model expressivity on prediction quality is not entirely clear. Structurally similar molecules can exhibit drastically different potencies towards the same protein target, a phenomenon known as activity cliffs (Pérez-Villanueva et al., 2015). The accurate identification of these activity cliffs is a challenging but essential task, as misclassifications may result in a drug candidate being erroneously interpreted as safe or toxic. Furthermore, distinguishing between stereoisomers is crucial, as these are molecules with the same molecular formula and bond structure but differ only in the three-dimensional spatial arrangement of their atoms. Even such small changes can significantly influence a molecule's biological activity and pharmacological profile.

The Lottery Ticket Hypothesis (LTH), originally proposed by Frankle & Carbin (2018), introduced the idea that large, randomly initialized neural networks contain smaller, train-

[1]Faculty of Computer Science, University of Vienna, Vienna, Austria [2]Doctoral School Computer Science, University of Vienna, Vienna, Austria [3]Research Network Data Science, University of Vienna, Vienna, Austria. Correspondence to: Lorenz Kummer <lorenz.kummer@univie.ac.at>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

able subnetworks, or *winning tickets*, that can match the performance of the full network. This hypothesis has gained considerable traction, especially in the context of deep learning, where iterative pruning techniques identify these subnetworks, significantly reducing model size and computational cost while preserving performance. LTH has since been extended to various domains, including GNNs. In GNNs, LTH has been explored primarily through the pruning of both graph structures (e.g., adjacency matrices) and model parameters (e.g., weights), leading to the discovery of Graph Lottery Tickets (GLTs), which maintain model performance while substantially reducing computational overhead (Chen et al., 2021a; Tsitsulin & Perozzi, 2023). Liu et al. (2024) provide a comprehensive survey on LTH and related works.

Despite extensive work on LTH in GNNs, and analogous investigations in related domains having yielded fundamental insights (Kummer et al., 2025), the link between LTH and GNN expressivity remains unexplored. While prior research focuses on efficiency, the effect of pre-training pruning on preserving expressivity is largely overlooked. Understanding this connection could reveal how sparse initialization impacts a GNN's ability to distinguish complex graph structures, and addressing this gap could advance more efficient graph learning models that maintain expressivity. Our work contributes to closing this gap by providing both formal and empirical evidence that preserving expressivity in sparsely initialized GNNs is crucial for finding winning tickets.

1.1. Related Work

LTH posits that large, randomly initialized neural networks contain smaller subnetworks, or winning tickets, that can match the full network's performance when trained in isolation (Frankle & Carbin, 2018). Through iterative pruning, these subnetworks are identified, significantly reducing the parameter count while preserving performance. Frankle et al. (2019) enhance LTH by introducing Iterative Magnitude Pruning (IMP) with rewinding, pruning subnetworks early in training rather than at initialization, thus finding sparse winning tickets in deep neural networks (DNNs) like ResNet-50, maintaining both accuracy and stability. Malach et al. (2020) extend LTH by proving the Strong Lottery Ticket Hypothesis (SLTH), showing over-parameterized networks contain subnetworks that achieve high accuracy without training, with theoretical guarantees for deep and shallow networks. Zhang et al. (2021a) theoretically explain LTH's improved generalization, showing pruned networks enlarge the convex region in the optimization landscape, enabling faster convergence and fewer samples for zero generalization error. Additionally, Zhang et al. (2021b) use Inertial Manifold Theory to validate LTH, showing that pruned subnetworks match dense network performance without repeated pruning and retraining. Finally, da Cunha et al. (2022) prove SLTH for CNNs, showing that large, randomly initialized CNNs can be pruned into subnetworks approximating fully trained models. Zhang et al. (2019) propose Eager Pruning, an LTH-based method that prunes DNNs early in training, reducing computation without accuracy loss. Their efficient hardware architecture achieves significant speedups and energy efficiency over Nvidia GPUs, suiting energy-constrained applications. Chen et al. (2021b) embed ownership verification into sparse subnetworks via graph-based signatures, resilient to fine-tuning and pruning attacks, enabling verification without performance impact.

In GNNs, LTH approaches typically treat pruning the graph and the GNN's trainable parameters as a unified task. Specifically, Chen et al. (2021a) introduce the Unified GNN Sparsification (UGS) framework, which prunes both graph adjacency matrices and model weights to identify GLTs and sparse subnetworks that maintain performance while reducing computational cost. Wang et al. (2022) propose transforming random subgraphs and subnetworks into GLTs through hierarchical sparsification and regularization-based pruning, achieving high performance with substantial sparsity. Tsitsulin & Perozzi (2023) present the GLT Hypothesis, suggesting that any graph contains a sparse substructure capable of preserving the performance of graph learning algorithms. Efficient algorithms are developed to find these GLTs, demonstrating that GNN performance is retained even on sparse subgraphs. Sui et al. (2023) introduce a co-pruning framework that prunes input graphs and model weights in both inductive and transductive settings, identifying GLTs while maintaining performance at high sparsity levels. Hui et al. (2023) improve the UGS framework by introducing an auxiliary loss for better edge pruning and a min-max optimization for robustness under high sparsity, enhancing pruning effectiveness. Zhang et al. (2024) present an automated adaptive pruning framework to identify GLTs in GNNs, optimizing graph and GNN sparsity without manual intervention and improving scalability in deeper GNNs. Yuxin et al. (2024) introduce a scalable graph structure learning method based on LTH, which prunes adjacency matrices and model weights to maintain performance under adversarial conditions. Finally, Yan et al. (2024) propose Multicoated Supermasks (M-Sup) and folding techniques to optimize GNNs based on SLTH, enhancing memory efficiency while maintaining performance.

1.2. Contribution

We formally link GNN expressivity to LTH by establishing criteria that pruning mechanisms—both graph and parameter pruning—must satisfy to preserve prediction quality. We demonstrate the existence of trainable subnetworks within moment-based GNNs that match 1-WL expressivity, putting forward the Strong Expressive Lottery Ticket Hypothesis (SELTH) as a novel, GNN-specific extension of the classical SLTH. We subsequently argue that critical computational

paths (i.e., those whose removal degrades performance) are subsets of these maximally expressive subnetworks. Furthermore, we show that expressive sparse initializations can improve generalization and convergence. We also identify cases where expressivity loss is irrecoverable, exemplified by molecular property prediction in a medical context. Finally, we empirically confirm that GNN parameter pruning impacts post-training accuracy and that more expressive sparse initializations are more likely to be winning tickets.

2. Preliminaries

A *graph* G is a pair (V, E) of a finite set of *nodes* V and *edges* $E \subseteq \{\{u, v\} \subseteq V\}$. The set of nodes and edges of G is denoted by $V(G)$ and $E(G)$, respectively. The *neighborhood* of a node v in $V(G)$ is $N(v) = \{u \in V(G) \mid \{u, v\} \in E(G)\}$. If a bijection $\varphi\colon V(G) \to V(H)$ with $\{u, v\} \in E(G) \iff \{\varphi(u), \varphi(v)\} \in E(H)$ for all $u, v \in V(G)$ exists, we call the two graphs G and H *isomorphic* and write $G \simeq H$. For two graphs with designated roots $r \in V(G)$ and $r' \in V(H)$, the bijection must further satisfy $\varphi(r) = r'$. The equivalence classes induced by $\simeq$ are referred to as *isomorphism types*. A function $l\colon V(G) \to \Sigma$ with an arbitrary codomain Σ is called a *node coloring*. Then, a *node colored* or *labeled graph* (G, l) is a graph G endowed with a node coloring l. We call $l(v)$ a *label* or *color* of $v \in V(G)$. For labeled graphs, the bijection φ must additionally satisfy $l(v) = l(v')$ if $\varphi(v) = v'$ for all $v \in V(G)$. We denote a multiset by $\{\!\{\ldots\}\!\}$.

The Weisfeiler-Leman algorithm. Let (G, l) denote a labeled graph. In every iteration $t > 0$, a node coloring $c_l^{(t)}\colon V(G) \to \Sigma$ (where the subscript indicates the coloring is based on the initial labeling function l) is computed, which depends on the coloring $c_l^{(t-1)}$ of the previous iteration. At the beginning, the coloring is initialized as $c_l^{(0)} = l$. In subsequent iterations $t > 0$, the coloring is updated according to $c_l^{(t)}(v) = \textsc{Hash}\left(c_l^{(t-1)}(v), \{\!\{c_l^{(t-1)}(u) | u \in N(v)\}\!\}\right)$, where HASH is an injective mapping of the above pair to a unique value in Σ, that has not been used in previous iterations. The HASH function can be implemented by assigning consecutive integers to pairs upon first occurrence (Shervashidze et al., 2011). Let $C_l^{(t)}(G) = \{\!\{c_l^{(t)}(v) \mid v \in V(G)\}\!\}$ be the multiset of colors a graph displays in iteration t. The iterative coloring terminates if $|C_l^{(t-1)}(G)| = |C_l^{(t)}(G)|$, i.e., the number of colors does not change between two iterations. For testing whether two graphs G and H are isomorphic, the above algorithm is run in parallel on both G and H. If $C_l^{(t)}(G) \neq C_l^{(t)}(H)$ for any $t \geq 0$, then G and H are not isomorphic. The label $c_l^{(t)}(v)$ in the tth iteration of the 1-WL test encodes the isomorphism type of the tree of height t representing v's t-hop neighborhood (D'Inverno et al., 2021; Jegelka, 2022; Schulz et al., 2022). We write $G \not\simeq_{\text{WL}^{(t)}} H$ to denote that 1-WL distinguishes G and H at iteration t.

From Deep Sets to graph neural networks. Deep Sets (Zaheer et al., 2017) model permutation-invariant functions, making them ideal for unordered or multi-instance data. Given a (multi)set $X = \{x_1, x_2, \ldots, x_n\}$, a function of the form $f(X) = \rho\left(\sum_{x \in X} \phi(x)\right)$ is by design invariant to the order of elements and can represent any such function for suitable choices of ρ and ϕ. These two functions, typically realized by multi-layer perceptrons (MLPs), map elements to a feature space, aggregate them and transform the result into the final output. This approach generalizes to multisets (Xu et al., 2019), reflecting both element identity and multiplicity, thus enabling permutation-invariant modeling of complex data distributions.

GNNs can be viewed as stacked neural multiset functions, where each layer aggregates and combines node features—numerical attributes assigned to nodes—reflecting the multiset nature of graph neighborhoods. This process, known as message passing (MP), applies moment functions derived from sum-pooling: $\hat{f}(\{\!\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}\!\}) = \sum_{i=1}^{k} f(\mathbf{x}_i)$ where $f\colon V^d \to V^m$ is an MLP mapping elements to a vector space (Amir et al., 2024). This allows unique representations for distinct multisets, as shown in Deep Sets. Applying this principle in GNNs, an MLP in each layer's update rule distinguishes different graph structures, effectively extending Deep Sets to graphs. The update rule of this class of *moment-based* GNNs employing this strategy can be generalized as in Equation (1), where $\mathbf{h}_v^{(k)}$ is the embedding of node v at layer k and $\mathbf{h}_v^{(0)}$ its initial feature vector. One-hot encoded input features ensure injectivity of summation at the first layer, even without an initial MLP. For graph-level tasks, the readout function aggregates outputs across layers to compute the graph embedding $\mathbf{h}_G$, see Equation (2), where $\|$ denotes concatenation.

$$\mathbf{h}_v^{(k)} = \text{MLP}^{(k)}\left(\mathbf{h}_v^{(k-1)} + \sum_{u \in N(v)} \mathbf{h}_u^{(k-1)}\right), \quad (1)$$

$$\mathbf{h}_G = \Big\|_{k=0}^{n} \sum_{v \in V(G)} \mathbf{h}_v^{(k)}. \quad (2)$$

The update rule can be expressed in matrix form using the adjacency matrix $\mathbf{A}$ and node feature matrix $\mathbf{H}$ as $\mathbf{H}^{(k)} = \text{MLP}^{(k)}\left((\mathbf{A} + \mathbf{I}) \cdot \mathbf{H}^{(k-1)}\right)$. The Graph Isomorphism Network (GIN) (Xu et al., 2019) extends neural multiset functions to graphs, ensuring distinct graph representations and achieving expressivity equal to the 1-WL test, the upper bound of MP-based GNNs. Unlike Equation (1) (and its matrix notation), GIN's update rule includes a learnable parameter $(1 + \epsilon^{(k)})$ multiplying the ego node embeddings

$\mathbf{h}_v^{(k-1)}$, distinguishing between the feature of the node and those of its neighbors, as captured in Lemma 2.1.

Lemma 2.1 (Sufficient condition for the 1-WL expressivity (Xu et al., 2019))**.** *A moment-based GNN $\Phi^{(k)}$ is as expressive as the 1-WL test after k iterations, if for all layers $j \in \{1, \ldots, k\}$ the function $\mathrm{MLP}^{(j)}$ is injective on its input domain and the aggregation rule distinguishes a node's own features from the aggregated features of its neighbors.*

Although a large body of work focuses on surpassing the expressivity of GIN and the 1-WL test (Morris et al., 2023), neighborhood aggregation remains prevalent, and 1-WL distinguishes most graphs in common benchmark datasets (Morris et al., 2021; Zopf, 2022).

3. Expressivity and Winning Tickets

We first establish fundamental criteria that any pruning mechanism, including those generating winning (graph) lottery tickets, must satisfy to maintain prediction quality. Let D be a finite sequence of tuples $(\mathbf{A}, \mathbf{X}, t)$, where graphs G are represented by adjacency and feature matrices $\mathbf{A}$ and $\mathbf{X}$, with classification targets t. The index $l \in [0, |D|]$ refers to the l^{th} tuple, denoted as $D_l = (\mathbf{A}, \mathbf{X}, t)$, with shorthand $G_l = D_l[\mathbf{A}, \mathbf{X}]$ and $t_l = D_l[t]$. Let $\Phi^{(k)}$ be an MPNN with k message-passing layers as defined in Equation (1), mapping graphs G_l to an embedding space Q. Consider $\Phi^{(k+1)}$, the same MPNN with an additional classification layer $\mathcal{C}$. Assume $\mathcal{C}$ is a perfect classifier that correctly assigns distinct embeddings from $\Phi^{(k)}$ to their respective classes, regardless of proximity in the embedding space. Specifically, for $G_a, G_b \in D$ with $t_a \neq t_b$ and $\Phi^{(k)}(G_a) \neq \Phi^{(k)}(G_b)$, it holds that: $t_a = \mathcal{C}(\Phi^{(k)}(G_a)) \neq \mathcal{C}(\Phi^{(k)}(G_b)) = t_b$. The existence of such a $\mathcal{C}$ is supported by the results of Chen et al. (2019), showing the equivalence between graph isomorphism testing and universal function approximation. For $\Phi^{(k+1)}$ to achieve the same prediction quality on a sequence of modified tuples $\widehat{D}$ (containing, e.g., $\widehat{D}_l = (\widehat{\mathbf{A}}, \mathbf{X}, t)$, where the l^{th} tuple's adjacency matrix has been pruned) as on D, this level of distinguishability must be preserved in pruned or otherwise adapted graphs. Similarly, any modification of $\Phi^{(k)}$ to $\widehat{\Phi}^{(k)}$ (such as pruning $\Phi^{(k)}$'s trainable parameters) must maintain this distinguishability for $\widehat{\Phi}^{(k+1)}$ to perform equivalently to $\Phi^{(k+1)}$:

Criterion 1. *For $\Phi^{(k+1)}$ (or $\widehat{\Phi}^{(k+1)}$) to classify all graphs in D (or $\widehat{D}$) correctly, $\Phi^{(k)}$ (or $\widehat{\Phi}^{(k)}$) must distinguish all pairs of non-isomorphic graphs of different classes.*

This requirement also applies to sparse initializations of the MLPs of the MP layers in $\Phi^{(k)}$, representing winning tickets in the initialization lottery, which we analyze for neural moment-based architectures. As our work focuses on such sparse initializations, the subsequent sections exclusively address this setting.

3.1. Critical Paths

We analyze the expressivity of neural moment-based architectures, as defined in Equation (1), by investigating the critical computational paths within the MLPs they contain. Let the *computational graph* of an L-layer feed-forward MLP be represented as $G = (V, E)$, where vertices V denote neurons and edges E their connections. Each neuron is indexed as $v^{(i_j)}$, referring to the i^{th} neuron in the j^{th} layer. A layer is thus a subgraph $G^{(j)} = (V^{(j)}, E^{(j)})$ with $V^{(j)} \subset V$ and $E^{(j)} \subset E$, characterized by the adjacency matrix $\mathbf{A}^{(j)} \in \{0,1\}^{|V^{(j)}| \times |V^{(j)}|}$. Defining input and output neurons as $I^{(j)} \subset V^{(j)}$ and $O^{(j)} \subset V^{(j)}$ such that $I^{(j)} \cup O^{(j)} = V^{(j)}$, the bipartite nature of $G^{(j)}$ implies that $\mathbf{A}^{(j)}$ is symmetric and can, thus, be expressed via the bi-adjacency matrix $\tilde{\mathbf{A}}^{(j)} \in \{0,1\}^{|I^{(j)}| \times |O^{(j)}|}$. The forward pass through the j^{th} layer is then $\sigma(\mathbf{x}\tilde{\mathbf{A}}^{(j)} \odot \mathbf{W}^{(j)})$, where $\mathbf{W}^{(j)}$ contains edge weights $\mathcal{W}^{(j)} = \{w_{k,l} \mid (v_k, v_l) \in E^{(j)}\}$, with the Hadamard product $\odot$ and activation function σ. Let $\mathcal{W}$ represent the set of all weights across layers. A path p from input neuron $v^{(i_0)}$ to output neuron $v^{(k_L)}$ consists of a sequence $(v^{(i_0)}, \ldots, v^{(k_L-1)})$, with edges $(v^{(i_j)}, v^{(k_{j+1})}) \in E$ for $j = 0, \ldots, L-1$. The total path length is L, and the set of all such paths is $\mathcal{P}$. Pruning in the MLP applies a binary mask $\mathbf{M}^{(j)}$ during the forward pass: $\sigma(\mathbf{x}(\mathbf{M}^{(j)} \odot \tilde{\mathbf{A}}^{(j)} \odot \mathbf{W}^{(j)}))$. Setting $\mathbf{M}^{(j)}_{k,l} = 0$ removes edge (v_k, v_l) from $E^{(j)}$, yielding $\widehat{E}^{(j)} = E^{(j)} \setminus \{(v_k, v_l)\}$. Consequently, the pruned path set $\widehat{\mathcal{P}}$ becomes: $\widehat{\mathcal{P}} = \mathcal{P} \setminus \{p \in \mathcal{P} \mid (v_k, v_l) \in p\}$, removing all paths relying on the pruned edge (v_k, v_l). These paths are crucial in GNN pruning, as GNNs rely on MLPs for transformations. In post-training pruning, where each MLP's weights $\mathcal{W}$ are fixed, a path is defined as *critical* if its removal degrades the GNN's classification performance. In the context of LTH, a path is critical if its removal prevents learning the task. Specifically, if no weight configuration for the remaining paths allows the GNN to perform as well as the original, the removed path is critical.

Definition 3.1 (Critical Paths (Pre-Training))**.** Let $\mathcal{W}_{\Phi^{(k)}} = \bigcup_i^k \mathcal{W}_i$ be the union of weights sets of the MLPs of the k MP layers of the moment-based GNN $\Phi^{(k)}$. Then, if for some quality metric $\mathcal{M}$ to be maximized and dataset D it holds that no set of weights $\mathcal{W}_{\widehat{\Phi}^{(k)}}$ exists such that $\mathcal{M}(\Phi^{(k+1)}, D) \leq \mathcal{M}(\widehat{\Phi}^{(k+1)}, D)$, we say $\mathcal{P}_{\Phi^{(k)},C} = \mathcal{P}_{\Phi^{(k)}} \setminus \mathcal{P}_{\widehat{\Phi}^{(k)}}$ is a critical path set.

The key question is which paths in GNNs are critical and how they are characterized. A path in a GNN is critical if its removal prevents the network from distinguishing two isomorphism types of different classes, regardless of the edge weights in any layer of any MLP. Any pruning mask that removes such a path (or an associated edge) cannot be a winning ticket, as it would degrade classification accu-

racy. Conversely, pruning that preserves all critical paths can, in theory—disregarding practical issues such as oversmoothing (Keriven, 2022), oversquashing (Di Giovanni et al., 2023), bottlenecks (Alon & Yahav, 2021) or vanishing gradients —constitute a winning ticket.

In the following theoretical considerations, we assume a class of injective continuously differentiable zero-fixing activations σ with a nowhere-zero derivative for simplicity. However, our empirical results indicate that our theory generalizes to arbitrary activations for appropriately parameterized GNNs, see Section 5. Due to the permutation invariance of GNNs, all theoretical results hold up to permutation. We do not assume a perfect classifier $\mathcal{C}$ for $\Phi^{(k+1)}$, unless stated otherwise. All proofs are provided in Appendix A.

Theorem 3.2 (Existence of maximally expressive paths)**.** *Let D be an arbitrary finite sequence of finite, non-trivial graphs (i.e., graphs with more than zero edges and at least one non-zero feature per node). Then, for any sufficiently overparameterized moment-based GNN $\Phi^{(k)}$ with layers employing an aggregation rule that can distinguish between a node's own features and the aggregated features of its neighbors, there exist subsets of maximally expressive paths $\mathcal{P}_{\Phi^{(k)},E} \subseteq \mathcal{P}_{\Phi^{(k)}}$ for which trainable weights $\mathcal{W}_{\widehat{\Phi}^{(k)},E}$ exist such that for any $G_a, G_b \in D$ it holds that if $G_a \not\simeq_{WL^{(k)}} G_b$ then $\widehat{\Phi}^{(k)}(G_a) \neq \widehat{\Phi}^{(k)}(G_b)$.*

While the formal results of SLTH (Malach et al., 2020; da Cunha et al., 2022) make the existence of maximally expressive paths seem plausible, they are not strictly equivalent. The proven SLTH variants are limited to MLPs or CNNs applied to classification tasks. Therefore, Theorem 3.2 constitutes a novel form of LTH, to which we refer as Strong Expressive Lottery Ticket Hypothesis or SELTH.

In line with Criterion 1, it immediately follows from Theorem 3.2 that for such a $\Phi^{(k)}$, a $\Phi^{(k+1)}$ employing a perfect classifier $\mathcal{C}$ could correctly classify every $G \in D$ and consequently $\mathcal{M}(\Phi^{(k+1)}, D) \leq \mathcal{M}(\widehat{\Phi}^{(k+1)}, D)$ for $\mathcal{P}_{\widehat{\Phi}^{(k)}} = \mathcal{P}_{\Phi^{(k)},E}$. Obviously, this existence statement is devoid of practical learning considerations, which we will address later. Before delving into those aspects, we first aim to explore how this result relates to the theory of critical paths. Specifically, it further follows directly from Theorem 3.2 that in every $\mathcal{P}_{\Phi^{(k)},E}$, there exist sufficiently expressive paths $\mathcal{P}_{\Phi^{(k)},S} \subseteq \mathcal{P}_{\Phi^{(k)},E}$ for which $\mathcal{W}_{\Phi^{(k)},S}$ exist such that for $G_a, G_b \in D$ it holds that if $t_a \neq t_b$ and $G_a \not\simeq_{\mathrm{WL}^{(k)}} G_b$ then $\Phi^{(k)}(G_a) \neq \Phi^{(k)}(G_b)$. This is an important insight, as by the definition of critical path sets, each critical path set is then a sufficiently expressive path set, and therefore, at least in a sufficiently overparameterized GNN, every critical path set is also a subset of a maximally expressive path set, for which Theorem 3.2 shows the existence.

3.2. Lottery Ticket Expressivity Impact on Training.

To examine the influence of expressivity on training, we consider its impact on gradient diversity (Yin et al., 2018), see Eq. (3). Originally devised for distributed training, gradient diversity has become a standard metric for assessing neural network learning in local settings as well (Kummer et al., 2023; Rajagopal et al., 2020). Specifically, low gradient diversity can slow convergence and necessitate more passes over the dataset to reach a desired accuracy. This increases training time, especially with large batch sizes, as the effective learning rate is diluted. Furthermore, models trained with low gradient diversity often converge to sharp minima, which are associated with poor generalization, particularly when using large batch sizes. Conversely, training with higher gradient diversity aligns with smoother loss surfaces and less sharp minima, improving the model's generalization, as well as convergence rates (Yin et al., 2018).

For graphs G_i with label t_i, the weight update steps in gradient-based training are given by $\mathbf{W}_i^{(l)} = \mathbf{W}^{(l)} - \alpha \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(l)}}$ for a learning rate α and a loss function loss $\mathcal{L}$. Gradient diversity is then given as follows for gradient matrices (using the Frobenius norm as a natural generalization of the Euclidean vector norm):

$$\Delta s = \left(\sum_{i=1}^{n} \left\| \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(l)}} \right\|_F^2 \right) \left(\left\| \sum_{i=1}^{n} \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(l)}} \right\|_F^2 \right)^{-1} \tag{3}$$

For this gradient diversity Δs, we formulate the following Theorem for moment-based GNNs, which captures the influence of the geometry of the embeddings on Δs's magnitude.

Theorem 3.3 (Gradient Diversity and Embeddings Orthogonality)**.** *Assume G_1, G_2 with labels $t_1 \neq t_2$ and let $\mathbf{H}_1^{(l-1)}, \mathbf{H}_2^{(l-1)}$ be their corresponding embeddings at layer $l-1$. Denote the rows of $\mathbf{H}_1^{(l-1)}$ as $\{\mathbf{a}_i\}$ and the rows of $\mathbf{H}_2^{(l-1)}$ as $\{\mathbf{b}_j\}$. Suppose there exist constants $0 < m \leq M < \infty$ s.t. for all i, j, $m \leq ||\mathbf{a}_i||_2^2, ||\mathbf{b}_j||_2^2 \leq M$. Then, there exists ζ s.t. $\Delta s \geq \zeta \geq 0$ with $\zeta \propto (\sum_{ij} |\cos(\beta_{ij})|)^{-1}$ with β_{ij} being the angle between rows $\mathbf{a}_i$ and $\mathbf{b}_j$.*

The theorem easily generalizes to any number of graphs (i.e., more than two). While the theorem specifically addresses the angles between individual node embeddings of two graphs, it also carries implications for expressivity. For instance, two graphs with identical node embeddings will always have at least as many codirectional embeddings as they have nodes (provided the graphs have the same number of nodes). Conversely, if the two graphs receive distinct node embeddings, their embeddings may still be codirectional but they can also be orthogonal, contributing zero to the sum of cosines, or otherwise divergent. Therefore, a model initialized such that two graphs do not receive (partially) identical node embeddings is likely to converge faster and generalize more effectively and thus more likely to be a winning

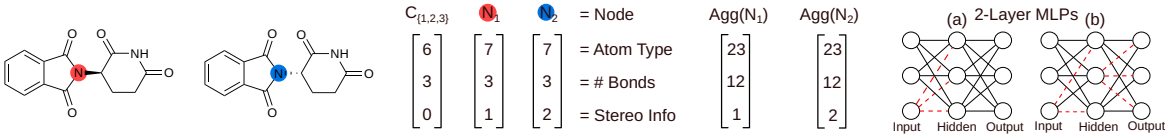


Figure 1. Visualization of thalidomide (also known as under its trade name Contergan) and its embryotoxic stereoisomer as an exemplary SIFDG, for which a failure to distinguish the two can have potentially life altering consequences for patients. As shown on the right-hand side of the figure, the aggregates of nodes N_1 and N_2 in the first layer differ by only a single feature. A pruning mask removing the red dashed edges from the MLPs applied to these aggregates would irrecoverably cancel this difference; hence these edges are part of critical path sets (see Definition 3.1) of a GNN trained to predict different classes for the two molecules as in, e.g., toxicity categorization tasks.

ticket in the initialization lottery, as identical embeddings are always codirectional.

In the context of Theorem 3.2, this suggests pruning a GNN $\Phi^{(k)}$ s.t. a maximally expressive path set $\mathcal{P}_{\Phi^{(k)},E}$ remains and initializing the weights $\mathcal{W}_{\widehat{\Phi}^{(k)},E}$ of this pruned $\widehat{\Phi}^{(k)}$ accordingly. Distinctness of node embeddings alone, though, does not rule out that they lie along the same line through the origin. However, depending on $\Phi^{(k)}$'s width, the following Proposition holds:

Proposition 3.4 (Non-Colinearity under Random Pruning)**.** *For a sufficiently overparameterized moment-based GNN $\Phi^{(k)}$ with weights drawn from a continuous distribution and a finite sequence D of finite input graphs, almost any random initialization combined with random pruning yields a configuration where no codirectional embeddings occur.*

Thus, the added constraint to Theorem 3.2 that for any two nodes i, j with pairwise distinguishable 1-WL colors it holds that $\widehat{\Phi}^{(k)}(G_a)_i \notin \{\eta_{ij}\widehat{\Phi}^{(k)}(G_b)_j : \eta_{ij} \in \mathbb{R}\setminus\{0\}\}$ is almost always satisfied for sufficiently overparameterized $\Phi^{(k)}$.

3.3. Edge Cases and Boundaries of Learnability

First, we outline GNN-specific scenarios where expressivity is permanently and irrecoverably lost through pruning, regardless of the amount of training, and highlight practical cases where such a misaligned pruning could have substantial consequences. Finally, we establish bounds on the prediction quality a GNN can achieve in such cases.

Irrecoverable cases. Whereas for general MP layers, a reduction in expressivity is associated with a reduction in gradient diversity and thus convergence speed (see Section 3.2), an irrecoverable loss of expressivity can for certain graphs occur if the first layer is pruned incorrectly.

Lemma 3.5 (Irrecoverable Loss of Expressivity)**.** *Consider two graphs, G_1 and G_2, with permutation-equivalent adjacency matrices but distinct feature matrices. If the pruning mask of the first layer of the MP layer's MLP—when applied in isolation—renders the graph representations indistinguishable, then no choice of weights can restore the ability to distinguish between the two graphs.*

In other words, for structurally isomorphic graphs that are only distinguishable by their node feature matrices, a pruning mask that cancels the differences in the messages before the update step in the first layer will—independent of the non-zero values of the MLP's weights—render both graphs' node embeddings indistinguishable at the layer's output. As they are structurally isomorphic, as a consequence they will remain indistinguishable for the rest of the forward pass, regardless of any subsequent transformations. In the context of Theorem 3.2, Lemma 3.5 implies the removal of a path from the maximally expressive path set and, depending on the class labels G_1 and G_2, is closely related to pruning of critical paths as by Definition 3.1 and in violation of Criterion 1. We emphasize that Lemma 3.5 is considering the first layer of the first MP layer's MLP exclusively. In deeper layers, the input embeddings change during training due to changing transformations in upper layers, and what may prune away the distinction during the first iteration of training might not pose a problem in subsequent iterations. However, as Figure 1 (b) shows, the trait captured by Lemma 3.5 can extend beyond the first MLP layer, and even an extension beyond the first MP layer is plausible. Moreover, expressivity is not entirely lost if G_1 and G_2 do not have permutation-equivalent adjacency matrices, since subsequent layers can still approximate 1-WL, albeit with a different feature matrix for these graphs.

In chemical datasets, structurally similar or identical graphs (i.e., adjacency matrices identical up to permutation) can have distinct input features, leading to vastly different chemical properties (see Figure 1). Failure to distinguish such graphs after the first layer can be critical. For example, toxic stereoisomers may appear identical to their non-toxic counterparts, such as thalidomide (Lenz & Knapp, 1962; Bösl, 2014). Non-stereoisomeric cases, like Ethanol and Ethanethiol—one a beverage ingredient, the other an odorizer used in gas warning systems—illustrate this further. An improperly pruned GNN may be unable to distinguish these graphs regardless of training. We call such graphs as Structurally Isomorphic, Feature-Divergent Graphs (SIFDGs).

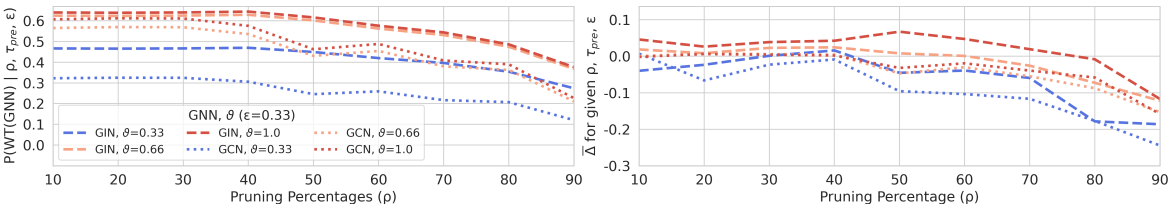


Figure 2. Probability that a sparse GIN or GCN is a winning ticket (WT), given pruning $\rho \in [10, 90]\%$, expressivity τ_{pre} s.t. $\vartheta - \varepsilon \leq \tau_{pre} \leq \vartheta + \varepsilon$, tolerance ε (left). ϑ and ε are used to group similar τ_{pre} into intervals, as observing an exact empirical value of τ_{pre} is unlikely. Mean relative test accuracy, $\overline{\Delta} = |S|^{-1} \sum_{i \in S} (A^{(i)}_{\text{post}} - A_{\text{clean}})(A_{\text{clean}})^{-1}$, where A_{clean} is clean test accuracy of the dense, unpruned model, $A^{(i)}_{\text{post}}$ is post-training accuracy of pruned model i, and S contains models with $\tau_{pre} \in [\vartheta \pm \varepsilon]$ (right). Results aggregated from training 13,500 runs over 10 datasets.

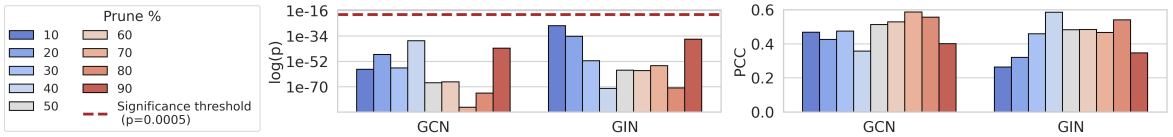


Figure 3. Statistical significance of Pearson correlation between untrained pruned model expressivity τ_{pre} ($\rho \in [10, 90]\%$, GIN, GCN) and post-training accuracy (left). Pearson correlation coefficients (PCC) (right). Aggregated over 13,500 runs (750 per bar, 10 datasets).

Theoretical bounds to the quality of a lottery ticket. Depending on the number of 1-WL distinguishable isomorphism types for which distinction in the first layer is necessary (i.e. SIFDGs) in order for the GNN to be able to learn to assign them to different classes, we can—under certain assumptions regarding class distributions—estimate how good a GNN can be if a pruning mask satisfying Lemma 3.5 is applied to the first layer.

Lemma 3.6. *Let D be a dataset of N graphs, evenly distributed across C classes of $\frac{N}{C}$ graphs each. Suppose D has I isomorphism types, of which U are indistinguishable from at least one other type by the model (i.e. $U \leq I$), covering M graphs. Assuming uniform distribution, $M \approx \frac{UN}{I}$, the maximum classification accuracy is:* $1 - \left(1 - \frac{1}{C}\right) \frac{U}{I}$.

If the U indistinguishable isomorphism types correspond to SIFDGs as derived from Lemma 3.5, then for any pruning mask satisfying the assumptions on data and class distribution outlined in Lemma 3.6, the above expression represents the maximal achievable accuracy for a such pruned GNN.

3.4. Generality and Limitations

Our theoretical results apply to moment-based GNN architectures (Section 2) in general and we thus expect the formal insights we developed—i.e., the connection between pruning, critical path removal, and loss of expressivity (e.g., Theorem 3.2, Lemma 3.5)—to apply broadly across this architectural class. This includes, for example, Graph Attention Network (Veličković et al., 2018) (GAT) or Graph Convolutional Network (GCN) (Welling & Kipf, 2016) as well. As such, we expect our upper bounds on achievable classification accuracy under misaligned pruning (e.g., Lemma 3.6) also hold for GAT or GCN. However, refining our formal analysis to architectures beyond this most general setting (including the effects of attention or edge features that modulate aggregation) is a promising direction for future work, which might reveal additional, architecture-specific vulnerabilities not covered by our existing work.

We furthermore emphasize that Lemma 3.6 is a theoretical bound requiring knowledge of all isomorphism types of a dataset, which is impractical—though tools like nauty[1] can identify them, and popular frameworks[2] include them for certain benchmark datasets. Most datasets likely also lack the assumed uniform class distribution. The lemma is meant to conceptually illustrate how misaligned pruning limits a GNN's maximal accuracy. Refining it for more realistic settings is a potential direction for future research.

We point out that Theorem 3.3 does not directly guarantee improved convergence or generalization but instead links embedding distinctiveness to gradient diversity, a factor known to influence both. Empirically, see Section 4, since all models were trained for the same number of epochs, the consistently superior performance of sparse initializations with high expressivity suggests improved convergence and generalization, which is consistent with our theory.

Our work focuses on graph level tasks, but we expect our findings to be transferable to node level tasks as well.

[1] https://pallini.di.uniroma1.it/
[2] https://pytorch-geometric

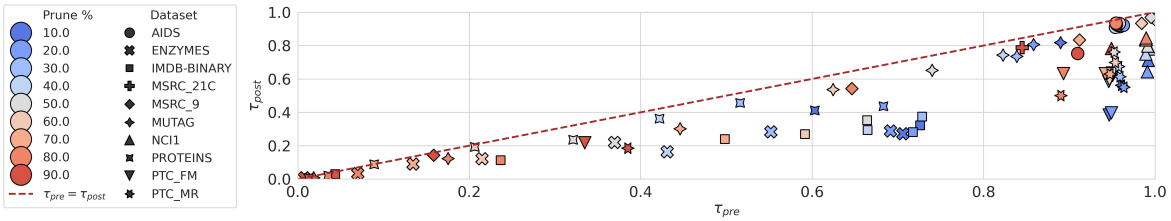


Figure 4. Visualization of untrained (τ_{pre}) and post-training (τ_{post}) expressivity of pruned models ($\rho \in [10, 90]\%$, GIN, GCN). Each marker represents the mean over the samples obtained for the models on the given dataset. The dashed line (0,0) to (1,1) indicates hypothetical pre-/post-training equivalence. Results based on 13,500 training runs in total over 10 datasets.

Table 1. Probability of post-training expressivity $\tau_{post} \geq \kappa$ given a pre-training expressivity $\tau_{pre} < \kappa$ for thresholds κ. Result based on 13,500 training runs in total (pruning percentages $\rho \in [10, 90]\%$, GIN, GCN) over 10 datasets.

$P(\tau_{post} \geq \kappa \mid \tau_{pre} < \kappa)$	0.00	0.03	0.04	0.03	0.04	0.03	0.03	0.03	0.03	0.03	0.02	0.02
κ	1.00	0.92	0.83	0.75	0.67	0.58	0.50	0.42	0.33	0.25	0.17	0.08

4. Experiments

We structure our experiments[3] to address the primary research question, which also drove our theoretical analysis, namely how the pre-training expressivity of a lottery ticket affects its post-training accuracy. That is, to what extent can we determine whether a sparsely initialized GNN is a winning lottery ticket based solely on its ability to distinguish non-isomorphic graphs? We investigate this research question by utilizing 10 real-world datasets from the TU-Dataset repository (Morris et al., 2020). These datasets, which are described in detail in Appendix B, are widely used in current studies. They include one-hot encoded node labels and span a variety of prevalent applications such as drug discovery (Xiong et al., 2021; Rossi et al., 2020), bioinformatics (Borgwardt et al., 2005), object recognition (Rossi & Ahmed, 2015), and social network analysis.

We assess our research question on GIN (2 hidden layers per MLP) with 2 to 4 MP layers due to its direct relation to our theory. To show our theory extends to GCN, representing the class of spectral GNNs, we include GCN in our assessment. We use ReLU activations and initialize network parameters $\mathbf{W}^{(j)}$ randomly from a uniform distribution $\mathcal{U}(-\sqrt{\frac{1}{m_j}}, \sqrt{\frac{1}{m_j}})$ with $m_j = |I^{(j)}|$, following common variance scaling initialization schemes (Glorot & Bengio, 2010; He et al., 2015). The parameterization width is matched with the input graphs' features. All models are trained for 250 epochs with a batch size of 32, a learning rate of 0.01, using the Adam optimizer. In line with LTH, only non-zero weights are updated. The experiments took approximately 8 weeks with three parallel workers to conclude and were conducted on a local server equipped with an NVIDIA H100 PCIe GPU (80GB VRAM), an Intel Xeon Gold 6326 CPU (500GB RAM) and a 1TB SSD.

We measure expressivity τ as graph level pre- and post-training expressivity (denoted τ_{pre} and τ_{post} in our experiments) via the percentage of non-isomorphic graphs of a dataset for which the GNN's final MP layer (at depth n) outputs node embeddings that are distinguishable for `FLOAT32` ($\epsilon_{mach} = 1.19 \times 10^{-7}$). Specifically, we retain one representative per isomorphism type, as isomorphic graphs yield identical embeddings by GNN permutation invariance. Let $\{G_1, \ldots, G_m\}$ be m such representatives with embeddings $\{\mathbf{h}_{G_1}, \ldots, \mathbf{h}_{G_m}\}$, $\mathbf{h}_{G_i} = \sum_{v \in V(G_i)} \mathbf{h}_v^{(n)}$. We mark each pair $(\mathbf{h}_{G_i}, \mathbf{h}_{G_j})$ as indistinguishable if $\mathbf{h}_{G_i} - \mathbf{h}_{G_j} = \mathbf{0}$, which generally implies differing node-level embeddings: while distinct node embeddings can theoretically sum to the same vector, such collisions are extremely rare and occur only under measure-zero configurations. The expressivity τ is the fraction that remains distinguishable. Alternative methods (e.g., exhaustive comparisons or training accuracy) are more costly or reflect different notions of expressivity. We chose our approach for its scalability and direct assessment of whether a graph is distinguishable from the rest of the dataset. A sparsely initialized model is a winning ticket if its test accuracy degrades by less than 0.05 relative to its dense counterpart.

5. Results

Our experiments empirically confirm both our theoretical predictions made in Section 3. Specifically, our experiments show that sparse yet highly expressive (even if their weights are untrained), trainable subnetworks exist (Theorem 3.2).

[3]The code for reproducing our results is available at GitHub: https://github.com/lorenz0890/wl2025lottery

Moreover, they show that for a given percentage of random weights pruning, the expressivity of the pruned network before training is at least one of the driving factors of its post-training accuracy.

The probability that a GNN is a winning ticket given a certain pruning percentage and pre-training expressivity is high for high expressivity and low otherwise for all but the highest pruning ratios (Figure 2, left). To compute this probability, for each pruning ratio ρ, we set a target ϑ and collect runs with $\tau_{pre} \in [\vartheta \pm \varepsilon]$. A run is labeled a "winning ticket" if its accuracy decreases by less than 0.05 relative to the unpruned model. The probability is the fraction of these runs, aggregated (with subset-size normalization) to visualize winning ticket probabilities across pruning levels and thresholds. Given that we trained all our models for the same number of epochs, our results are consistent with the hypothesis put forward in Section 3.2 that an increased expressivity in the initialization potentially improves model convergences and generalization, as is indicated by Theorem 3.3. Up to 80% pruning, highly expressive sparse GINs outperformed dense unpruned models in post-training prediction, while less expressive models failed to match dense model quality (Figure 2, right).

Moreover, we find that a GNN, when initialized with a certain sparsity and non-zero weights trained in isolation, is highly unlikely to gain expressivity (Table 1). That is, if a GNN initialized, pruned and trained as described could reliably transition from low τ_{pre} to high τ_{post}, then for some κ, this transition would occur with a probability exceeding a low single-digit percentage. Although we do not explicitly set a threshold defining "high probability", Figure 4 illustrates the trend outlined in Table 1. Consistent with Table 1, Figure 4 shows that sparse GNNs rarely transition from low to high expressivity during training. Instead, the converse is the typical case, as all data points fall below the dashed $\tau_{post} = \tau_{pre}$ line, indicating that τ_{post} is generally lower than τ_{pre}, whereby low pruning percentages apparently exaggerate the effect. This aligns with Lemma 3.5 (focused on the input layer) and highlights the broader trend suggested in Section 3.3: despite the dataset-dependent differences apparent in Figure 4, recovery from a relatively low expressivity sparse initialization is rare during training if only non-zero weights are updated, even if theoretically possible. As pre-training expressivity was low even at conservative pruning rates on some datasets, we interpret this, following Proposition 3.4, as an indication that the underlying GNNs were insufficiently parameterized for the pruning rate.

Finally, Figure 3 reveals an intriguing pattern: the statistical significance of the correlation between pre-training expressivity and post-training prediction quality (left) is lowest for mid to slightly above mid-pruning ratios but higher for very low and high pruning ratios (though still significant). Conversely, the Pearson correlation coefficients (right) show an inverse trend. We conjecture this reflects the greater likelihood of irrecoverable damage (compare Section 3.3) at high pruning ratios, while achieving good post-training prediction quality is generally more probable at low pruning ratios and less related to expressivity, leading to increased p-values and decreased correlation coefficients in both cases.

Implications for practitioners and future research. As demonstrated in this work, the expressivity of sparsely initialized GNNs influences convergence speed and generalization quality. Irrecoverable pruning scenarios, where degraded expressivity cannot be restored through training, underscore the need for careful pruning design. Preserving critical paths is essential to avoid catastrophic errors, such as misclassifying toxic and non-toxic stereoisomers.

Future work should seek to enhance convergence, generalization, and optimization by developing sparse yet expressive initializations. For moment-based GNNs, injective aggregate and combine functions over dataset-defined domains are sufficient for maximal expressivity (i.e., 1-WL -equivalence for GIN). A simple pre-training sparsification strategy could proceed layer-wise: for a fixed pruning ratio, sample k configurations and retain the one preserving input–output injectivity across all graphs. This is repeated per layer, increasing sparsity until no injective configuration is found within k trials. The result is a maximally expressive, sparsified network. Adapting our results on winning tickets to settings such as those explored by Wałęga & Rawson (2025), Bause et al. (2024) or Kummer et al. (2024) is another promising direction. Pruning strategies that prioritize expressivity could improve LTH's practicality for GNNs, ensuring reliability and performance in critical domains.

6. Conclusion

In this work, we bridge the gap between GNN expressivity and LTH. We offer theoretical insights and empirical validation that trainable sparse initializations with comparable expressivity to dense models exist. Moreover, we show that increased expressivity in the initialization potentially accelerates model convergence and improves generalization. Our findings highlight the risk of pruning strategies that fail to retain critical computational paths, in certain cases theoretically causing irreversible degradation in performance. Recovering expressivity loss induced by sparsity through training is generally unlikely, underscoring the need for sparsification approaches that safeguard key expressive capabilities. Our work also has practical implications: preserving critical paths during pruning is essential to prevent catastrophic errors, such as misclassifying toxic and non-toxic stereoisomers, particularly in critical applications like drug discovery or molecular property prediction.

Acknowledgements

This work was supported in part by the Vienna Science and Technology Fund (WWTF) and the City of Vienna through grants [10.47379/VRG19009] and [10.47379/ICT22059].

Impact Statement

Our work highlights the critical role of preserving expressivity in sparsely initialized and trained GNNs, offering insights into how pruning can lead to irrecoverable expressivity loss. By demonstrating the risks associated with degraded expressivity, we show the importance of careful pruning design to ensure reliable predictions, such as distinguishing toxic from non-toxic compounds in drug discovery. Moreover, via linking GNN expressivity with generalization and convergence, we show that degraded expressivity can hinder both training dynamics and model reliability. The broader societal benefit of our work lies in its potential to inform the development of high-performance and trustworthy GNNs in resource-constrained settings, contributing to equitable access to advanced machine learning technologies and promoting their safe use in impactful domains.

References

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. *International Conference on Learning Representations (ICLR)*, 2021.

Amir, T., Gortler, S., Avni, I., Ravina, R., and Dym, N. Neural injective functions for multisets, measures and graphs via a finite witness theorem. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 42516–42551, 2024.

Bause, F., Moustafa, S., Langguth, J., Gansterer, W. N., and Kriege, N. M. On the two sides of redundancy in graph neural networks. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, pp. 371–388, 2024.

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S., Smola, A. J., and Kriegel, H.-P. Protein function prediction via graph kernels. *Bioinformatics*, 21:47–56, 2005.

Bösl, E. The contergan scandal. media, medicine, and thalidomide in 1960s west germany. *Disability Histories*, pp. 136–162, 2014.

Chen, T., Sui, Y., Chen, X., Zhang, A., and Wang, Z. A unified lottery ticket hypothesis for graph neural networks. In *International Conference on Machine Learning (ICML)*, pp. 1695–1706, 2021a.

Chen, X., Chen, T., Zhang, Z., and Wang, Z. You are caught stealing my winning lottery ticket! making a lottery ticket claim its ownership. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1780–1791, 2021b.

Chen, Z., Villar, S., Chen, L., and Bruna, J. On the equivalence between graph isomorphism testing and function approximation with gnns. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 15894 – 15902, 2019.

Cheung, M. and Moura, J. M. Graph neural networks for covid-19 drug discovery. In *IEEE International Conference on Big Data*, pp. 5646–5648, 2020.

da Cunha, A., Natale, E., and Viennot, L. Proving the strong lottery ticket hypothesis for convolutional neural networks. In *International Conference on Learning Representations (ICLR)*, 2022.

Di Giovanni, F., Giusti, L., Barbero, F., Luise, G., Lio, P., and Bronstein, M. M. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In *International Conference on Machine Learning (ICML)*, pp. 7865–7885, 2023.

D'Inverno, G. A., Bianchini, M., Sampoli, M. L., and Scarselli, F. A unifying point of view on expressive power of GNNs. *CoRR*, abs/2106.08992, 2021.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Stabilizing the lottery ticket hypothesis. *CoRR*, abs/1903.01611, 2019.

Gao, J., Lyu, T., Xiong, F., Wang, J., Ke, W., and Li, Z. Predicting the survival of cancer patients with multimodal graph neural network. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 19(2):699–709, 2022.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 249–256, 2010.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *International conference on computer vision (ICCV)*, pp. 1026–1034, 2015.

Hui, B., Yan, D., Ma, X., and Ku, W.-S. Rethinking graph lottery tickets: Graph sparsity matters. In *International Conference on Learning Representations (ICLR)*, 2023.

Jegelka, S. Theory of graph neural networks: Representation and learning. *CoRR*, abs/2204.07697, 2022.

Keriven, N. Not too little, not too much: a theoretical analysis of graph (over) smoothing. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:2268–2281, 2022.

Kummer, L., Sidak, K., Reichmann, T., and Gansterer, W. N. Adaptive precision training (AdaPT): A dynamic quantized training approach for DNNs. In *SIAM International Conference on Data Mining (SDM)*, pp. 559–567, 2023.

Kummer, L., Moustafa, S., Schrittwieser, S., Gansterer, W. N., and Kriege, N. M. Attacking graph neural networks with bit flips: Weisfeiler and Leman go indifferent. In *International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD)*, 2024.

Kummer, L., Gansterer, W. N., and Kriege, N. M. On the relationship between robustness and expressivity of graph neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2025.

Lenz, W. and Knapp, K. Thalidomide embryopathy. *Archives of Environmental Health: An International Journal*, 5(2):14–19, 1962.

Liu, B., Zhang, Z., He, P., Wang, Z., Xiao, Y., Ye, R., Zhou, Y., Ku, W.-S., and Hui, B. A survey of lottery ticket hypothesis. *CoRR*, abs/2403.04861, 2024.

Lu, H. and Uddin, S. A weighted patient network-based framework for predicting chronic diseases using graph neural networks. *Scientific reports*, 11(1):22607, 2021.

Malach, E., Yehudai, G., Shalev-Schwartz, S., and Shamir, O. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning (ICML)*, pp. 6682–6691, 2020.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020.

Morris, C., Fey, M., and Kriege, N. The power of the weisfeiler-leman algorithm for machine learning with graphs. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4543–4550, 2021.

Morris, C., Lipman, Y., Maron, H., Rieck, B., Kriege, N. M., Grohe, M., Fey, M., and Borgwardt, K. Weisfeiler and leman go machine learning: The story so far. *Journal of Machine Learning Research*, 24(333):1–59, 2023.

Neumann, M., Garnett, R., Bauckhage, C., and Kersting, K. Propagation kernels: efficient graph kernels from propagated information. *Machine learning*, 102:209–245, 2016.

Pérez-Villanueva, J., Méndez-Lucio, O., Soria-Arteche, O., and Medina-Franco, J. L. Activity cliffs and activity cliff generators based on chemotype-related activity landscapes. *Molecular Diversity*, 19(4):1021–1035, 2015.

Rajagopal, A., Vink, D., Venieris, S., and Bouganis, C.-S. Multi-precision policy enforced training (MuPPET) : A precision-switching strategy for quantised fixed-point training of CNNs. In *International Conference on Machine Learning (ICML)*, pp. 7943–7952, 2020.

Rossi, R. and Ahmed, N. The network data repository with interactive graph analytics and visualization. In *Conference on Artificial Intelligence (AAAI)*, 2015.

Rossi, R. A., Jin, D., Kim, S., Ahmed, N. K., Koutra, D., and Lee, J. B. On proximity and structural role-based embeddings in networks: Misconceptions, techniques, and applications. *ACM Transactions on Knowledge Discovery from Data*, 14(5):1–37, 2020.

Schomburg, I., Chang, A., and Schomburg, D. Brenda, enzyme data and metabolic information. *Nucleic acids research*, 30(1): 47–49, 2002.

Schulz, T. H., Horváth, T., Welke, P., and Wrobel, S. A generalized weisfeiler-lehman graph kernel. *Machine Learning*, 111(7): 2601–2629, 2022.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

Sui, Y., Wang, X., Chen, T., Wang, M., He, X., and Chua, T.-S. Inductive lottery ticket learning for graph neural networks. *Journal of Computer Science and Technology*, 2023.

Sun, Z., Yin, H., Chen, H., Chen, T., Cui, L., and Yang, F. Disease prediction via graph neural networks. *IEEE Journal of Biomedical and Health Informatics*, 25(3):818–826, 2021.

Tsitsulin, A. and Perozzi, B. The graph lottery ticket hypothesis: Finding sparse, informative graph structure. *CoRR*, abs/2312.04762, 2023.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Wałęga, P. A. and Rawson, M. Expressive power of temporal message passing. In *Conference on Artificial Intelligence (AAAI)*, volume 39, pp. 21000–21008, 2025.

Wang, K., Liang, Y., Wang, P., Wang, X., Gu, P., Fang, J., and Wang, Y. Searching lottery tickets in graph neural networks: A dual perspective. In *International Conference on Learning Representations (ICLR)*, 2022.

Welling, M. and Kipf, T. N. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2016.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Xiong, J., Xiong, Z., Chen, K., Jiang, H., and Zheng, M. Graph neural networks for automated de novo drug design. *Drug Discovery Today*, 26(6):1382–1393, 2021.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

Yan, J., Ito, H., García-Arias, Á. L., Okoshi, Y., Otsuka, H., Kawamura, K., Van Chu, T., and Motomura, M. Multicoated and folded graph neural networks with strong lottery tickets. In *Learning on Graphs Conference (LoG)*, pp. 11–1, 2024.

Yanardag, P. and Vishwanathan, S. Deep graph kernels. In *ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 1365–1374, 2015.

Yin, D., Pananjady, A., Lam, M., Papailiopoulos, D., Ramchandran, K., and Bartlett, P. Gradient diversity: a key ingredient for scalable distributed learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1998–2007, 2018.

Yuxin, W., Xiannian, H., Jiaqing, X., Zhangyue, Y., Yunhua, Z., Xipeng, Q., and Xuanjing, H. Graph structure learning via lottery hypothesis at scale. In *Asian Conference on Machine Learning*, pp. 1401–1416, 2024.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Póczos, B., Salakhutdinov, R., and Smola, A. J. Deep sets. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 3391–3401, 2017.

Zhang, G., Wang, K., Huang, W., Yue, Y., Wang, Y., Zimmermann, R., Zhou, A., Cheng, D., Zeng, J., and Liang, Y. Graph lottery ticket automated. In *International Conference on Learning Representations (ICLR)*, 2024.

Zhang, J., Chen, X., Song, M., and Li, T. Eager pruning: Algorithm and architecture support for fast training of deep neural networks. In *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, pp. 292–303, 2019.

Zhang, S., Wang, M., Liu, S., Chen, P.-Y., and Xiong, J. Why lottery ticket wins? a theoretical perspective of sample complexity on sparse neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2707–2720, 2021a.

Zhang, Z., Jin, J., Zhang, Z., Zhou, Y., Zhao, X., Ren, J., Liu, J., Wu, L., Jin, R., and Dou, D. Validating the lottery ticket hypothesis with inertial manifold theory. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 30196–30210, 2021b.

Zopf, M. 1-WL expressiveness is (almost) all you need. In *International Joint Conference on Neural Networks, (IJCNN)*, pp. 1–8, 2022.

A. Proofs

A.1. Proof of Theorem 3.2

Proof. The proof consists of two parts. In the first part, we show the existence of subsets paths for which weights exist such that any 1-WL distinguishable pair of graphs of a bounded dataset can be distinguished. In the second part, we show that the weights of these paths are trainable, i.e., that they can receive gradient updates.

As established in the literature (Lemma 2.1), a neural moment-based GNN is maximally expressive if all of its MLPs are injective on their respective input domains; that is, if distinct inputs representing neighborhood aggregates are mapped to distinct outputs. Therefore, it is necessary to demonstrate the existence of sparse subnetworks capable of modeling such injective functions. Noting that an MLP is injective if all of its layers are injective, we begin by proving the following Lemma:

Lemma A.1 (Layer Injectivity under Random Pruning)**.** *Let $\mathbf{W} \in \mathbb{R}^{m\times n}$ with $1 < n \leq m$ be a linear transformation matrix where each entry is independently drawn from a continuous and bounded distribution over an interval $[a, b]$ with $a < b$. Let $\sigma : \mathbb{R} \to \mathbb{R}$ an injective elementwise activation function. Consider a finite inpute set $X \subset \mathbb{R}^n$ with cardinality N where each element has continous entries bounded by $[c, d]$. Suppose we randomly prune $\mathbf{W}$ by independently setting each weight to zero with probability ρ (sparsity ratio), resulting in a sparse weight matrix $\mathbf{W}'$. Then, the function $f = \sigma \circ \mathbf{W}' : X \to \mathbb{R}^m$ is injective with probability $\gamma \geq 1 - \binom{N}{2}\rho^{km}$ where $k \geq 1$ is the minimum number of non-zero components in $\mathbf{x}_u - \mathbf{x}_v$ for all distinct pairs $\mathbf{x}_u, \mathbf{x}_v \in X$*

Proof. The function f fails to be injective if there exist distinct inputs $\mathbf{x}_u, \mathbf{x}_v \in X$ s.t. $f(\mathbf{x}_u) = f(\mathbf{x}_v)$. Since σ is injective, this implies that $\mathbf{W}'\mathbf{x}_u = \mathbf{W}'\mathbf{x}_v$ Let $\delta = \mathbf{x}_u - \mathbf{x}_v \neq \mathbf{0}$. Then, the condition $\mathbf{W}'\mathbf{x}_u = \mathbf{W}'\mathbf{x}_v$ becomes $\mathbf{W}'\delta = \mathbf{0}$.

For a fixed δ, we compute the probability that $\mathbf{W}'\delta = \mathbf{0}$. Let k be the number of non-zero components in δ, i.e., $k = \|\delta\|_0 \geq 1$. Each weight w'_{rj} in $\mathbf{W}'$ is then given by $w'_{rj} = w_{rj}z_{rj}$ where w_{rj} is the original weights and z_{rj} is an independent Bernoulli random variable:

$$z_{rj} = \begin{cases} 0, & \text{with probability } \rho, \\ 1, & \text{with probability } 1-\rho. \end{cases}$$

For each row r of $\mathbf{W}'$, the r-th component of $\mathbf{W}'\delta$ is:

$$(\mathbf{W}'\delta)_r = \sum_{j=1}^{n} w'_{rj}\delta_j = \sum_{j=1}^{n} w_{rj}z_{rj}\delta_j.$$

If all $z_{rj}\delta_j = 0$**:** This occurs if either $z_{rj} = 0$ (due to pruning) or $\delta_j = 0$.

- The probability that $z_{rj} = 0$ when $\delta_j \neq 0$ is ρ.
- Since z_{rj} are independent and there are k non-zero δ_j, the probability that all $z_{rj} = 0$ for those j is ρ^k.

If any $z_{rj}\delta_j \neq 0$**:** Since w_{rj} are continuous random variables over $[a, b]$ and $\delta_j \neq 0$, the product $w_{rj}\delta_j$ is a continuous random variable. Therefore, the sum $(\mathbf{W}'\delta)_r$ is also a continuous random variable, and the probability that it equals zero is zero. That is, the set of weights elements of each row vector $\mathbf{w}_r$ satisfying $\mathbf{w}_r^T\delta = 0$ forms a hyperplane in $\mathbb{R}^n$. Since elements w_{rj} are drawn independently from a continuous distribution, the probability that they fall on that hyperplane is practically 0, which is equivalent to every proper hyperplane having a zero Lebesgue measure in its ambient space.

Thus, the probability $(\mathbf{W}'\delta)_r = 0$ is

$$P\left((\mathbf{W}'\delta)_r = 0\right) = \rho^k.$$

Therefore, as the rows of $\mathbf{W}'$ are independent because the w'_{rj} are independent across r, the probability that $\mathbf{W}'\delta = \mathbf{0}$ is

$$P\left(\mathbf{W}'\delta = \mathbf{0}\right) = \left(\rho^k\right)^m = \rho^{km}.$$

There are $\binom{N}{2}$ distinct pairs $(\mathbf{x}_u, \mathbf{x}_v)$ in X. By Boole's inequality, the probability that there exists at least one pair $(\mathbf{x}_u, \mathbf{x}_v)$ such that $\mathbf{W}'\delta = \mathbf{0}$ is now at most:

$$P\left(\exists \delta \neq \mathbf{0} : \mathbf{W}'\delta = \mathbf{0}\right) \leq \binom{N}{2}\rho^{km}.$$

Therefore, the probability that f is injective over X is at least:

$$\gamma = 1 - \binom{N}{2}\rho^{km}.$$

□

Since $k \geq 1$ and $\rho \in (0,1)$, ρ^{km} decreases exponentially with m, and for sufficiently large m, γ will always be > 0. That is, for $\rho > 0, \gamma > 0$ and for fixed N, k, we choose m such that $m \geq \log_\rho \left((1-\gamma)\binom{N}{2}^{-1}\right) k^{-1}$.

Then, it immediately follows that for an MLP with L layers, the probability of a random pruning leaving it injective is $\gamma_L \geq (1 - \binom{N}{2}\rho^{km_{min}})^L$ with m_{min} being the smallest number of neurons of any of the L layers. Note that treating the layers independently is conservative, as non-injectivity in a previous layer would decrease N (size of distinct elements in input) for the subsequent layer and thus increase the probability of it being injective w.r.t to these inputs.

Likewise, we can model a GNN on a bounded dataset D of finite graphs. If there are at most N nodes in any graph in D and we have M MP layers, the total probability for the pruned GNN having injective MLPs is then $\gamma_{GNN} \geq (1 - \binom{|D|N}{2}\rho^{km_{min}})^{LM}$, which corresponds to the probability that a non-zero pruning rate (corresponding to pruning masks corresponding to edge- and therefore computational path-deletions) leaves a subset of paths intact in the GNN for which weights exist such that the GNN is maximally expressive (in the sense of Lemma 2.1) on the given dataset D, i.e., if $G_a \not\simeq_{WL^{(k)}} G_b, G_a, G_b \in D$ then $\widehat{\Phi}^{(k)}(G_a) \neq \widehat{\Phi}^{(k)}(G_b)$, given a sufficient number of neurons m for each of the MLP layers.

Based on the above shown existence of $\mathcal{P}_{\widehat{\Phi}^{(k)},E} \subseteq \mathcal{P}_{\Phi^{(k)}}$ and $\mathcal{W}_{\widehat{\Phi}^{(k)},E}$ such that any $G_a, G_b \in D, G_a \not\simeq_{WL^{(k)}} G_b$ can be distinguished via $\widehat{\Phi}^{(k)}(G_a) \neq \widehat{\Phi}^{(k)}(G_b)$, we now show that these weights can receive weights updates. For simplicity and to focus on the most relevant interactions between trainable parameters and other components of MP, the proof only considers MP layers with a single layer MLP (i.e., not hidden layers), but it can easily be generalized to an arbitrary number of layers per MLP.

Lemma A.2 (Trainability)**.** *Let $\widehat{\Phi}^{(k)}$ be a GNN pruned to an $\mathcal{P}_{\widehat{\Phi}^{(k)},E}$ initialized to appropriate $\mathcal{W}_{\widehat{\Phi}^{(k)},E}$ for a dataset D, where D consists of bounded, non-trivial graphs (e.g., graphs with more than zero edges and at least one non-zero feature per node) Then, for any loss $\mathcal{L}(\widehat{\Phi}^{(k+1)}(G), t) \neq 0$, at least those weights relevant to the distinction of G from other $H \in D$ are capable of receiving updates.*

Proof. The general equations for backpropagation through a GNN with $\mathbf{Z}^{(l)} = \mathbf{A}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}$ and $\mathbf{H}^{(l)} = \sigma(\mathbf{Z}^{(l)})$ of the form Equation (1), where $\frac{\partial \mathbf{Z}^{(l)}}{\partial \mathbf{H}^{(l)}} = \mathbf{W}^{(l)T}$, $\frac{\partial \mathcal{L}}{\partial \mathbf{H}^{(l)}} = \mathbf{A}^T \frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{(l+1)}} \mathbf{W}^{(l+1)T}$, and $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}}$ are derived via chain rule as

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} &= \frac{\partial \mathcal{L}}{\partial \mathbf{H}^{(l)}} \frac{\partial \mathbf{H}^{(l)}}{\partial \mathbf{W}^{(l)}} \\
&= \frac{\partial \mathcal{L}}{\partial \mathbf{H}^{(l)}} \frac{\partial \mathbf{H}^{(l)}}{\partial \mathbf{Z}^{(l)}} \frac{\partial \mathbf{Z}^{(l)}}{\partial \mathbf{W}^{(l)}} \\
&= \mathbf{H}^{(l-1)T}\mathbf{A}^T \frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{(l)}} \\
&= \mathbf{H}^{(l-1)T}\mathbf{A}^T \left(\sigma'(\mathbf{Z}^{(l)}) \odot \frac{\partial \mathcal{L}}{\partial \mathbf{H}^{(l)}}\right) \\
&= \mathbf{H}^{(l-1)T}\mathbf{A}^T \left(\sigma'(\mathbf{Z}^{(l)}) \odot \mathbf{A}^T \frac{\partial \mathcal{L}}{\partial \mathbf{Z}^{(l+1)}} \mathbf{W}^{(l+1)T}\right)
\end{aligned} \tag{4}$$

which, written as elementwise summation, yields

$$\frac{\partial \mathcal{L}}{\partial W_{ij}^{(l)}} = \sum_{l=1}^{d_l} \sum_{r=1}^{n} \sum_{p=1}^{n} \sum_{q=1}^{n} H_{qi}^{(l-1)} A_{pq} A_{rp} \sigma'(Z_{pj}^{(l)}) \frac{\partial \mathcal{L}}{\partial Z_{rl}^{(l+1)}} W_{jl}^{(l+1)} \tag{5}$$

To show back-propagation works for these paths, we show by induction the existence of non-zero gradients at every layer.

Base Case. By assumption $\widehat{\Phi}^{(k)}$ is pruned to an expressive path set $\mathcal{P}_{\widehat{\Phi}^{(k)},E}$ initialized to appropriate $\mathcal{W}_{\widehat{\Phi}^{(k)},E}$, allowing the distinction of WL distinguishable graphs. That means for any G in D, there must be at least one node embedding at the k^{th} layers output of $\widehat{\Phi}^{(k)}$ that distinguishes it from those $H \in D$ that are $H \not\simeq_{WL^{(k)}} G$, as otherwise either $\mathcal{P}_{\widehat{\Phi}^{(k)},E}$ would not be a maximally expressive path set or $\mathcal{W}_{\widehat{\Phi}^{(k)},E}$ would not be properly initialized as per Theorem 3.2. That is, at least one of the two graphs G or H must have at least one non-zero node embeddings and, for the sake of the argument, we let this graph be G. Let's further say p is one of those nodes of G where embeddings are distinguishing G from H.Then, this node p must have at least one non-zero embedding i at layer k, i.e., $\sigma(Z_{pi}^{(k)}) \neq 0$,

Now, we start at layer $l = k$, right before the classifier layer. Assume $\frac{\partial \mathcal{L}}{\partial Z_{rt}^{(k+1)}} W_{it}^{(k+1)} \neq 0$, i.e. loss back propagated for the t^{th} feature of the r^{th} node from the classifier $\mathcal{C}$ used at layer $k+1$ and the t^{th} weight of the i^{th} neuron of the classifier is also non-zero (which it is since we assume a dense classifier with random continuous weights).

Then we know by assumption $\sigma(Z_{pi}^{(k)}) \neq 0$ that for some node p connected to r (or at least $r = p$) at least one feature i of node p must have had a non-zero contribution to $\sigma(Z_{pi}^{(k)}) \neq 0$ as σ is injective and zero-fixing (i.e. only $\sigma(0) = 0$) and consequently, as σ has nowhere-zero derivative, at least for these indices,

$$\frac{\partial \mathcal{L}}{\partial Z_{pi}^{(k)}} = \sigma'(Z_{pi}^{(k)}) \sum_{t=1}^{d_k} \sum_{r=1}^{n} A_{rp} \frac{\partial \mathcal{L}}{\partial Z_{rt}^{(k+1)}} W_{it}^{(k+1)} \neq 0. \tag{6}$$

From the assumption $\sigma(Z_{pi}^{(k)}) \neq 0$ and σ being zero-fixing it further follows that

$$Z_{pi}^{(k)} = \sum_{q}^{n} \sum_{j}^{d_{k-1}} A_{pq} H_{qj}^{(k-1)} W_{ji}^{(k)} \neq 0 \tag{7}$$

which implies that for some node q connected to r (or $r = q$) the j^{th} feature had a non-zero embedding at layer $k-1$, i.e., $H_{qj}^{(k-1)} \neq 0$, and moreover, $W_{ji}^{(k)} \neq 0$ (which in turn implies inclusion of the edge ji of the computational graph in the expressive path set $\mathcal{P}_{\widehat{\Phi}^{(k)},E}$).

Thus, for the gradient update step,

$$\frac{\partial \mathcal{L}}{\partial W_{ji}^{(k)}} = \sum_{t=1}^{d_l} \sum_{r=1}^{n} \sum_{p=1}^{n} \sum_{q=1}^{n} H_{qj}^{(k-1)} A_{pq} A_{rp} \sigma'(Z_{pi}^{(k)}) \frac{\partial \mathcal{L}}{\partial Z_{rt}^{(k+1)}} W_{it}^{(k+1)} \tag{8}$$

we find that indices exist which for which the non-zero weight $W_{ji}^{(k)}$ as part of the expressive path set receives a nonzero gradient and the loss gradient $\frac{\partial \mathcal{L}}{\partial Z_{pi}^{(k)}}$ that can be back propagated to the next layer.

Assumption. For some su and iu $\frac{\partial \mathcal{L}}{\partial Z_{su}^{(l+1)}} W_{iu}^{(l+1)} \neq 0$.

Step. In the induction step, we show that the induction assumption implies that the error is back propagated through layer $l-1$ and the weights at layer $l-1$ receive updates via the recursive definition of backpropagation:

$$\frac{\partial \mathcal{L}}{\partial W_{ji}^{(l-1)}} = \sum_{t=1}^{d_{l-1}} \sum_{r=1}^{n} \sum_{p=1}^{n} \sum_{q=1}^{n} \sum_{u=1}^{d_l} \sum_{s=1}^{n} H_{qj}^{(l-2)} A_{pq} A_{rp} A_{sr} \sigma'(Z_{pi}^{(l-1)}) \sigma'(Z_{rt}^{(l)}) \frac{\partial \mathcal{L}}{\partial Z_{su}^{(l+1)}} W_{iu}^{(l+1)} W_{it}^{(l)} \tag{9}$$

By the induction assumption, for some su and iu, $\frac{\partial \mathcal{L}}{\partial Z_{su}^{(l+1)}} W_{iu}^{(l+1)} \neq 0$. As we also assume that $\sigma(Z_{pi}^{(k)}) \neq 0$ for some pi, also $\sigma(Z_{rt}^{(l)}) \neq 0$ for some rt as otherwise, no non-zero forward propagation could have happened. By the same argument, for some pi, $\sigma(Z_{pi}^{(l-1)}) \neq 0$, implying $\frac{\partial \mathcal{L}}{\partial Z_{pi}^{(l-1)}} \neq 0$ for some node p. Thus, assuming $W_{ji}^{(l-1)} \neq 0$, as the edge is part the expressive path set, it follows that $H_{qj}^{(l-2)} \neq 0$ for some qj. Then, if nodes pq, rp and sr are connected or have self loops, at least for those node indices, $\frac{\partial \mathcal{L}}{\partial W_{ji}^{(l-1)}} \neq 0$. □

To summarize, we have shown that it follows from Lemma A.1 that for any arbitrary finite sequence of finite graphs D (e.g., graphs with more than zero edges and at least one non-zero feature per node). and any sufficiently overparameterized moment-based GNN $\Phi^{(k)}$ with layers employing an aggregation rule that can distinguish between a node's own features and the aggregated features of its neighbors, there exist subsets of maximally expressive paths $\mathcal{P}_{\Phi^{(k)},E} \subseteq \mathcal{P}_{\Phi^{(k)}}$ for which weights $\mathcal{W}_{\widehat{\Phi}^{(k)},E}$ exist such that for any $G_a, G_b \in D$ it holds that if $G_a \not\simeq_{WL^{(k)}} G_b$ then $\widehat{\Phi}^{(k)}(G_a) \neq \widehat{\Phi}^{(k)}(G_b)$ and, from Lemma A.2, that these $\mathcal{W}_{\widehat{\Phi}^{(k)},E}$ for $\mathcal{P}_{\Phi^{(k)},E} \subseteq \mathcal{P}_{\Phi^{(k)}}$ are trainable. □

A.2. Proof of Theorem 3.3

Proof. Let G_i with labels t_i be graphs and $\mathcal{L}_i$ be the loss w.r.t to the input graph G_i with label t_i. Then, gradient diversity as given by (3) via Frobenius inner product expansion, can be rewritten as

$$\Delta s = \Big(\sum_{i=1}^{n} || \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(l)}} ||_F^2 \Big) \Big(\sum_{i=1}^{n} || \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(l)}} ||_F^2 + \sum_{i \neq j} \Big\langle \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(l)}}, \frac{\partial \mathcal{L}_j}{\partial \mathbf{W}^{(l)}} \Big\rangle_F \Big)^{-1}. \tag{10}$$

Obviously, the Frobenius inner product (a generalization of the dot product to matrices) between gradients of different data points dictates how large or small the gradients' diversity is: if all gradients are orthogonal , it is maximal, whereas if they are codirectional, it is minimal.

Let's now consider two graphs specific G_1, G_2 with labels $t_1 \neq t2$ and analyze this inner product product. Then, with

$$\frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(l)}} = \mathbf{H}_i^{(l-1)T} \mathbf{A}_i^T \frac{\partial \mathcal{L}}{\partial \mathbf{Z}_i^{(l)}} \tag{11}$$

from Equation (4) we obtain for the inner product

$$tr\Big(\Big(\mathbf{H}_1^{(l-1)T} \mathbf{A}_1^T \frac{\partial \mathcal{L}_1}{\partial \mathbf{Z}_1^{(l)}} \Big)^T \mathbf{H}_2^{(l-1)T} \mathbf{A}_2^T \frac{\partial \mathcal{L}_2}{\partial \mathbf{Z}_2^{(l)}} \Big) = \Big\langle \mathbf{H}_1^{(l-1)T} \mathbf{A}_1^T \frac{\partial \mathcal{L}_1}{\partial \mathbf{Z}_1^{(l)}}, \mathbf{H}_2^{(l-1)T} \mathbf{A}_2^T \frac{\partial \mathcal{L}_2}{\partial \mathbf{Z}_2^{(l)}} \Big\rangle_F \tag{12}$$

which (via transposition and cyclicity of the trace of matrix products) is equal to

$$tr\Big(\mathbf{H}_1^{(l-1)} \mathbf{H}_2^{(l-1)T} \mathbf{A}_2^T \frac{\partial \mathcal{L}_2}{\partial \mathbf{Z}_2^{(l)}} \frac{\partial \mathcal{L}_1}{\partial \mathbf{Z}_1^{(l)}} \mathbf{A}_1 \Big) = tr\Big(\Big(\mathbf{H}_1^{(l-1)T} \mathbf{A}_1^T \frac{\partial \mathcal{L}_1}{\partial \mathbf{Z}_1^{(l)}} \Big)^T \mathbf{H}_2^{(l-1)T} \mathbf{A}_2^T \frac{\partial \mathcal{L}_2}{\partial \mathbf{Z}_2^{(l)}} \Big). \tag{13}$$

For this left hand side, without assumptions of definiteness of the matrices involved, an upper bound of it's magnitude (i.e. absolute value) is provided by

$$|tr\Big(\mathbf{H}_1^{(l-1)} \mathbf{H}_2^{(l-1)T} \mathbf{A}_2^T \frac{\partial \mathcal{L}_2}{\partial \mathbf{Z}_2^{(l)}} \frac{\partial \mathcal{L}_1}{\partial \mathbf{Z}_1^{(l)}} \mathbf{A}_1 \Big)| \leq ||\mathbf{H}_1^{(l-1)} \mathbf{H}_2^{(l-1)T}||_F \cdot ||\mathbf{A}_2^T \frac{\partial \mathcal{L}_2}{\partial \mathbf{Z}_2^{(l)}} \frac{\partial \mathcal{L}_1}{\partial \mathbf{Z}_1^{(l)}} \mathbf{A}_1||_F. \tag{14}$$

The first part of this product itself can be bounded by exploiting the relation between the Frobenius norm and the inner product

$$||\mathbf{H}_1^{(l-1)} \mathbf{H}_2^{(l-1)T}||_F \leq \sqrt{(\max_{i,j} ||\mathbf{a}_i||_2^2 ||\mathbf{b}_j||_2^2) \sum_{i,j} \cos^2(\beta_{ij})} \tag{15}$$

with i, j denoting rows of $\mathbf{H}_1^{(l-1)}, \mathbf{H}_2^{(l-1)}$ and $\mathbf{a}_i, \mathbf{b}_j$ the corresponding row vectors and β_{ij} the angle between each pair $\mathbf{a}_i, \mathbf{b}_j$ and $\sum_{i,j}$ the double sum over the common dimension of the two matrices. Thus, it follows that if all node embeddings are orthogonal, gradient diversity is maximal as their enclosing angles' cosines equal 0.

Furthermore, if there exist m, M s.t. for all i, j it holds that $m \leq ||\mathbf{a}_i||_2^2, ||\mathbf{b}_j||_2^2 \leq M$, then $\max_{i,j} ||\mathbf{a}_i||_2^2 ||\mathbf{b}_j||_2^2 \leq M^2$ and thus $||\mathbf{H}_1^{(l-1)} \mathbf{H}_2^{(l-1)T}||_F \leq M\sqrt{\sum_{i,j} \cos^2(\beta_{ij})}$, which is proportional up to a constant factor M to the sum of the cosine similarity of the node embeddings.

Consequentially, we can formulate for the two graphs G_1, G_2

$$\zeta_{\pm} = \Big(\sum_{i=1}^{2} || \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(l)}} ||_F^2 \Big) \Big(\sum_{i=1}^{2} || \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(l)}} ||_F^2 \pm M \sqrt{\sum_{i,j} \cos^2(\beta_{ij})} \cdot || \mathbf{A}_2^T \frac{\partial \mathcal{L}_2}{\partial \mathbf{Z}_2^{(l)}} \frac{\partial \mathcal{L}_1}{\partial \mathbf{Z}_1^{(l)}} \mathbf{A}_1 ||_F \Big)^{-1}. \tag{16}$$

for which it holds that for any $\sum_{i=1}^{2} || \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(l)}} ||_F^2 \neq 0$ either $\Delta s \in [\zeta_+, \infty)$ if $0 < \sum_{i=1}^{2} || \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(l)}} ||_F^2 < M\sqrt{\sum_{i,j} \cos^2(\beta_{ij})} \cdot || \mathbf{A}_2^T \frac{\partial \mathcal{L}_2}{\partial \mathbf{Z}_2^{(l)}} \frac{\partial \mathcal{L}_1}{\partial \mathbf{Z}_1^{(l)}} \mathbf{A}_1 ||_F$ or $\Delta s \in [\zeta_+, \zeta_-]$ if $\sum_{i=1}^{2} || \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}^{(l)}} ||_F^2 \geq M\sqrt{\sum_{i,j} \cos^2(\beta_{ij})} \cdot || \mathbf{A}_2^T \frac{\partial \mathcal{L}_2}{\partial \mathbf{Z}_2^{(l)}} \frac{\partial \mathcal{L}_1}{\partial \mathbf{Z}_1^{(l)}} \mathbf{A}_1 ||_F$.

Thus, for any Δs, we obtain the proposed lower bound by choosing $\zeta = \zeta_+$ for which it directly follows from equation (16) that $\zeta \propto (\sum_{ij} |cos(\beta_{ij})|)^{-1}$. □

A.3. Proof of Proposition 3.4

Proof. Let the function

$$f = \sigma \circ \mathbf{W}' : X \to \mathbb{R}^m,$$

where $\mathbf{W} \in \mathbb{R}^{m \times n}$, $1 < n \leq m$, with entries independently drawn from a continuous and bounded distribution over an interval $[a, b]$ with $a < b$. We use an injective elementwise activation $\sigma : \mathbb{R} \to \mathbb{R}$. Suppose we randomly prune $\mathbf{W}$ by independently setting each weight to zero with probability ρ, resulting in a sparse matrix $\mathbf{W}'$. Let $X \subset \mathbb{R}^n$ be a finite input set of cardinality N, with each element having continuous entries bounded by $[c, d]$.

Then, by Lemma A.1, with probability

$$\gamma \geq 1 - \binom{N}{2} \rho^{k\,m},$$

the map $f = \sigma(\mathbf{W}' \cdot)$ is injective on X, where $k \geq 1$ is the minimum number of non-zero components in $\mathbf{x}_u - \mathbf{x}_v$ for all distinct pairs $\mathbf{x}_u, \mathbf{x}_v \in X$. We also recall from the argument in the proof of Theorem 3.2 that, for $\rho > 0$, $\gamma > 0$, and fixed N, k, one obtains a sufficient condition on m, namely $m \geq \log_\rho \left((1 - \gamma) \binom{N}{2}^{-1} \right) k^{-1}$

We now extend Lemma A.1 from injectivity to *non-colinearity* of the outputs. Specifically, we wish to show that for any two *distinct* inputs $\mathbf{x}_u, \mathbf{x}_v \in X$, the probability that there exists a nonzero scalar $\eta \in \mathbb{R}$ with

$$f(\mathbf{x}_u) = \eta f(\mathbf{x}_v)$$

is negligible (indeed measure zero) under our assumptions on the continuous distribution of weights and the independent pruning process.

We proceed as follows. Fix a single pair of distinct inputs $\mathbf{x}_u \neq \mathbf{x}_v \in X$. Denote by $\mathbf{w}'_r$ the r-th row of $\mathbf{W}'$. Then

$$f(\mathbf{x}_u) = \eta f(\mathbf{x}_v) \quad \iff \quad \sigma({\mathbf{w}'_r}^T \mathbf{x}_u) = \eta \sigma({\mathbf{w}'_r}^T \mathbf{x}_v) \quad \text{for all} r = 1, \ldots, m.$$

Because σ is injective (elementwise), each equation

$$\sigma({\mathbf{w}'_r}^T \mathbf{x}_u) = \eta \sigma({\mathbf{w}'_r}^T \mathbf{x}_v)$$

for a *fixed* η imposes a lower-dimensional (measure-zero) constraint on the row $\mathbf{w}'_r$ in $\mathbb{R}^n$. Enforcing this condition *across all* $r = 1, \ldots, m$ and with the *same* scalar η yields an intersection of these measure-zero sets in $\mathbb{R}^{m \times n}$. Consequently, the probability that

$$\mathbf{W}' \in \left\{ \mathbf{W}' \in \mathbb{R}^{m \times n} : f(\mathbf{x}_u) = \eta f(\mathbf{x}_v) \right\}$$

is itself zero under our continuous distribution for $\mathbf{W}$ and subsequent Bernoulli pruning of entries (c.f. Lemma A.1). The only way the dimension of these constraints would fail to be negligible is if $\mathbf{W}'$ had some degenerate structure (e.g. all-zero rows, or precisely tuned rows to force colinearity), but for almost all choices of non-pruned weights in a sufficiently

overparameterized $\mathbf{W}'$ (i.e. with $m \geq \log_\rho \left((1-\gamma)\binom{N}{2}^{-1}\right) k^{-1}$ for $\gamma \to 1$), such a degeneracy cannot occur with non-zero probability.

Hence, for any *fixed* pair $\mathbf{x}_u, \mathbf{x}_v \in X$, the probability that they become mapped to colinear outputs under f is zero. A union bound (Boole's inequality) over the $\binom{N}{2}$ distinct pairs in X remains a finite union of measure-zero events. Therefore, with probability 1, *none* of the pairs $\mathbf{x}_u, \mathbf{x}_v \in X$ yield colinear outputs.

Putting it all together:

- *Injectivity*: we already know from Lemma A.1 that $P[f(\mathbf{x}_u) \neq f(\mathbf{x}_v) \forall \mathbf{x}_u \neq \mathbf{x}_v] \geq 1 - \binom{N}{2}\rho^{km}$.
- *Non-colinearity*: with probability 1, no two distinct inputs map to outputs that are scalar multiples of each other.

Since a measure-zero event does not further reduce the probability threshold from the injectivity part, the second property is effectively guaranteed *almost surely* in our random draw of $\mathbf{W}'$. Therefore, for any two distinct points in X, both

$$f(\mathbf{x}_u) \neq f(\mathbf{x}_v) \quad \text{and} \quad f(\mathbf{x}_u) \not\propto f(\mathbf{x}_v)$$

hold with high probability (and indeed the colinearity condition holds with probability 0).

Thus, if m is chosen large enough to satisfy $m \geq \log_\rho \left((1-\gamma)\binom{N}{2}^{-1}\right) k^{-1}$ then $\gamma \geq 1 - \binom{N}{2}\rho^{km} > 0$, ensuring both injectivity and non-colinearity of the outputs as claimed. □

A.4. Proof of Lemma 3.5

Proof. Let G_1 and G_2 be two graphs with adjacency and features matrices $\mathbf{A}_1 = \mathbf{A}_2$ and $\mathbf{X}_1 \neq \mathbf{X}_2$, respectively. Let $\mathbf{W}^{(1)}$ be the first weight's matrix of the first MP layers MLP and $\mathbf{M}^{(1)}$ it's associated pruning mask. Suppose $\mathbf{A}_1\mathbf{X}_1\mathbf{M}^{(1)} = \mathbf{A}_2\mathbf{X}_2\mathbf{M}^{(1)}$. Then, we need to show that for any weight matrix $\mathbf{W}^{(1)}$, $\mathbf{A}_1\mathbf{X}_1\mathbf{W}^{(1)} = \mathbf{A}_2\mathbf{X}_2\mathbf{W}^{(1)}$.

Now, assume, for contradiction, that there exists a weight matrix $\mathbf{W}^{(1)}$ such that $\mathbf{A}_1\mathbf{X}_1\mathbf{W}^{(1)} \neq \mathbf{A}_2\mathbf{X}_2\mathbf{W}^{(1)}$. Consider the elementwise product $\mathbf{M}^{(1)} \odot \tilde{\mathbf{A}}^{(1)} \odot \mathbf{W}^{(1)}$. Since $\mathbf{M}^{(1)}$ and $\tilde{\mathbf{A}}^{(1)}$ are binary matrices, this product selects certain entries of $\mathbf{W}^{(1)}$.

If $\mathbf{A}_1\mathbf{X}_1\mathbf{W}^{(1)} \neq \mathbf{A}_2\mathbf{X}_2\mathbf{W}^{(1)}$, there must exist some i, j for which

$$\langle (\mathbf{A}_1\mathbf{X}_1)_{i,:}, \mathbf{W}^{(1)}_{:,j} \rangle \neq \langle (\mathbf{A}_2\mathbf{X}_2)_{i,:}, \mathbf{W}^{(1)}_{:,j} \rangle.$$

Since we can focus on the entries selected by $\mathbf{M}^{(1)} \odot \tilde{\mathbf{A}}^{(1)}$, we consider

$$\langle (\mathbf{A}_1\mathbf{X}_1)_{i,:}, (\mathbf{M}^{(1)} \odot \tilde{\mathbf{A}}^{(1)} \odot \mathbf{W}^{(1)})_{:,j} \rangle \quad \text{and} \quad \langle (\mathbf{A}_2\mathbf{X}_2)_{i,:}, (\mathbf{M}^{(1)} \odot \tilde{\mathbf{A}}^{(1)} \odot \mathbf{W}^{(1)})_{:,j} \rangle.$$

If these two inner products differ, then there exists at least one l, l' such that

$$(\mathbf{A}_1\mathbf{X}_1)_{i,l}(\mathbf{M}^{(1)}_{l,j}\tilde{\mathbf{A}}^{(1)}_{l,j}\mathbf{W}^{(1)}_{l,j}) \neq (\mathbf{A}_2\mathbf{X}_2)_{i,l'}(\mathbf{M}^{(1)}_{l',j}\tilde{\mathbf{A}}^{(1)}_{l',j}\mathbf{W}^{(1)}_{l',j}).$$

For this difference to depend on $\mathbf{W}^{(1)}$, the corresponding entries of $\mathbf{M}^{(1)}$ and $\tilde{\mathbf{A}}^{(1)}$ must be nonzero. In particular, if there is a discrepancy when using $\mathbf{W}^{(1)}$, then choosing $\mathbf{W}^{(1)}$ such that it matches the nonzero structure selected by $\mathbf{M}^{(1)}$ and $\tilde{\mathbf{A}}^{(1)}$ would produce the same discrepancy. Hence, a difference in $\mathbf{A}_1\mathbf{X}_1\mathbf{W}^{(1)}$ and $\mathbf{A}_2\mathbf{X}_2\mathbf{W}^{(1)}$ would imply a difference in $\mathbf{A}_1\mathbf{X}_1\mathbf{M}^{(1)}$ and $\mathbf{A}_2\mathbf{X}_2\mathbf{M}^{(1)}$, contradicting the initial assumption.

Thus, our contradiction shows that no such $\mathbf{W}^{(1)}$ can exist. Therefore, for any $\mathbf{W}^{(1)}$, it must hold that $\mathbf{A}_1\mathbf{X}_1\mathbf{W}^{(1)} = \mathbf{A}_2\mathbf{X}_2\mathbf{W}^{(1)}$. □

A.5. Proof of Lemma 3.6

Proof. Under the assumption that classes are uniformly represented, each of the C classes contains about $\frac{M}{C}$ of these M indistinguishable graphs, and the pairwise indistinguishability of isomorphism types of different classes is equivalent to

assigning all M graphs to a single class. Then, the model correctly classifies exactly those $\frac{M}{C}$ graphs that truly belong to the chosen class. The other $M - \frac{M}{C} = M\left(1 - \frac{1}{C}\right)$ graphs from the remaining $C - 1$ classes are misclassified.

For the remaining $N - M$ graphs, which are all distinguishable isomorphism types, the model can correctly classify all of them (assuming perfect separability in the distinguishable subset).

Hence, the total number of correctly classified graphs is

$$(N - M) + \frac{M}{C}.$$

Dividing by N to obtain the accuracy:

$$\frac{(N - M) + \frac{M}{C}}{N} = 1 - \frac{M}{N} + \frac{M}{CN}.$$

Since $M \approx \frac{UN}{I}$, substitute this into the equation:

$$= 1 - \frac{U}{I} + \frac{U}{IC}.$$

Factor out $\frac{U}{I}$:

$$= 1 - \frac{U}{I}\left(1 - \frac{1}{C}\right).$$

This expression gives the maximum fraction of correctly classified graphs under the stated assumptions.

□

B. DATASETS

Table 2. Overview of selected datasets.

DATASET	TYPE	#GRAPHS	AVG. NODES	AVG. EDGES	LABELS	TASK
MUTAG	CHEMICAL	188	17.9	19.8	NODE, EDGE	CLASS.
AIDS	CHEMICAL	2,000	15.7	16.2	NODE, EDGE	CLASS.
PTC_FM	CHEMICAL	349	14.1	14.5	NODE, EDGE	CLASS.
PTC_MR	CHEMICAL	344	14.3	14.7	NODE, EDGE	CLASS.
NCI1	CHEMICAL	4,110	29.9	32.3	NODE	CLASS.
PROTEINS	PROTEIN	1,113	39.1	72.8	NODE	CLASS.
ENZYMES	PROTEIN	600	32.6	62.1	NODE	CLASS.
MSRC_9	IMAGE	221	40.6	72.4	NODE	CLASS.
MSRC_21C	IMAGE	563	77.5	142.8	NODE	CLASS.
IMDB-BINARY	SOCIAL	1,000	19.8	96.5	-	CLASS.

To examine the complex relationship between the Lottery Ticket Hypothesis (LTH) and the expressivity of Graph Neural Networks (GNNs), we utilize a diverse set of ten real-world datasets. Each dataset is carefully selected based on its relevance to distinct graph structures and domain-specific tasks. These datasets are sourced from the widely recognized TUDataset collection (Morris et al., 2020), which serves as a standard benchmark for tasks involving graph classification and regression. A detailed summary of these datasets is presented in Table 2.

Chemical Compounds: The MUTAG, AIDS, PTC_FM, PTC_MR, and NCI1 datasets focus on chemical compounds, where molecular structures are represented as graphs, with nodes corresponding to atoms and edges denoting chemical bonds. MUTAG, one of the earliest and most widely used datasets for graph classification, comprises nitroaromatic compounds labeled by their mutagenic effects on Salmonella typhimurium. The AIDS dataset contains molecular graphs relevant to anti-HIV drug discovery, aiming to predict inhibitory activity against HIV. The PTC datasets (FM and MR) involve compounds evaluated for rodent carcinogenicity, classified based on different experimental setups. NCI1, a larger dataset derived from the National Cancer Institute's screening program, involves classifying compounds according to their activity against non-small cell lung cancer.

Protein Structures: The PROTEINS and ENZYMES datasets are derived from bioinformatics, where graphs are used to represent protein structures. In the PROTEINS dataset, nodes correspond to secondary structural elements, such as helices and sheets, with edges reflecting spatial proximity. The classification task is to determine whether a given protein functions as an enzyme. The ENZYMES dataset builds on this by categorizing enzymes into one of the six top-level classes defined by the Enzyme Commission (EC), based on the types of chemical reactions they catalyze (Borgwardt et al., 2005; Schomburg et al., 2002).

Image Segmentation: The MSRC_9 and MSRC_21C datasets, originating from the MSRC database, are designed for semantic image segmentation tasks. In these datasets, images are represented as graphs, where nodes correspond to superpixels and edges capture spatial relationships between them. The task requires GNNs to classify nodes into various categories based on the visual content of the superpixels. MSRC_21C serves as an extended version, offering a greater number of classes and increased complexity (Neumann et al., 2016).

Social Networks: The IMDB-BINARY dataset represents social networks, with each graph modeling the collaboration network of actors who have appeared together in movies. The classification task involves distinguishing these networks based on the movie genre, specifically differentiating between action and romance. This dataset highlights the complexities of real-world social networks, where nodes correspond to individuals and edges represent their interactions, challenging GNNs to capture nuanced social dynamics (Yanardag & Vishwanathan, 2015).

These datasets collectively offer a diverse evaluation framework, encompassing a wide variety of graph structures and complexities, ranging from small molecular graphs to larger social networks and image-based graphs. Leveraging this extensive set of benchmarks allows our empirical analysis to thoroughly evaluate the hypothesis across multiple domains.