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 $\varphi\colon V(G) \to V(H)$ 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 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), $\lfloor \ldots \rceil$ 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 $\widehat{\ldots}$ 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,

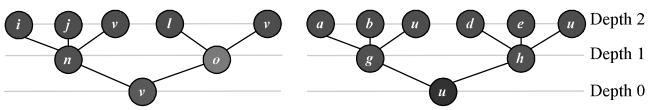


Figure 1: Example of two non-isomorphic unfolding trees $T^{(2)}(u) \not\simeq T^{(2)}(v)$ of height 2 associated with the nodes u and v. A function solving a WL-discrimination task for $k = 2$ must be able to discriminate u and v based on the structure of their unfolding trees.

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].

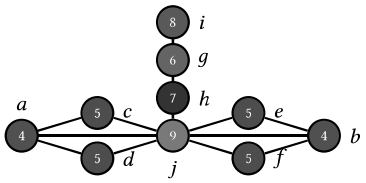
We investigate the functions of a GIN layer and their contribution to the expressivity of the final output, guiding the formulation of effective attacks. This requires determining whether our focus is on the expressivity of the general function on nodes or graphs in inductive learning or distinguishing 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 $\mathbf{Y}_b$.

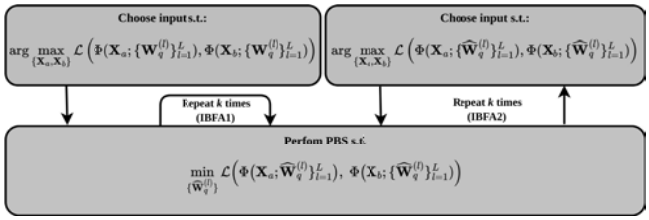


Figure 3: IBFA1/2's integration of PBS and input data selection strategies for k attack iterations. In the first attack iteration, input data selection of IBFA1 and IBFA2 are identical.

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

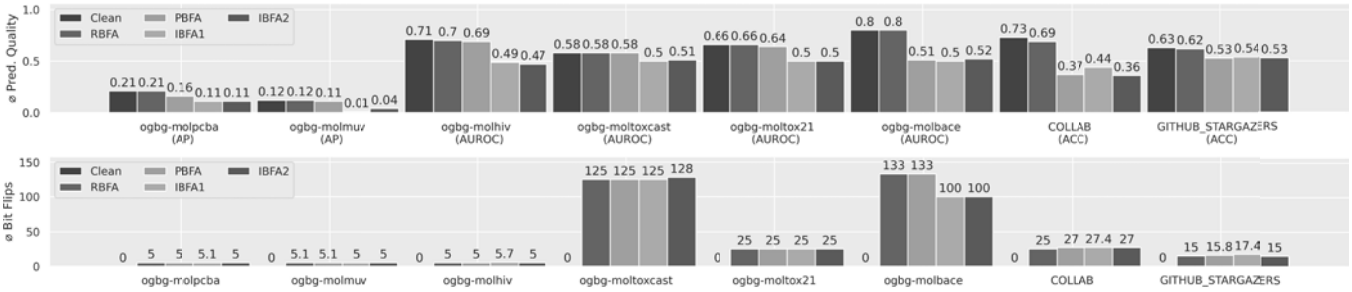


Figure 4: Pre- (clean) and post-attack test quality metrics AP, AUROC or ACC for different BFA variants on a 5-layer GIN trained on six `ogbg-mol` and two TUDataset datasets, number of bit flips, averages of 10 runs.

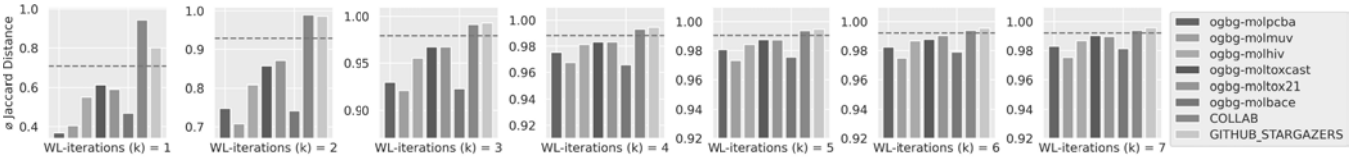


Figure 5: Jaccard distances $J_m(C_l^{(k)}(G_i), C_l^{(k)}(G_j)), i \neq j$, averaged over all graphs G_i, G_j and tasks (classes) in each randomly drawn sample of 3200 graphs per dataset, indicating ε-GLWL-discrimination tasks. The dotted lines distinguish molecular and social datasets for various numbers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

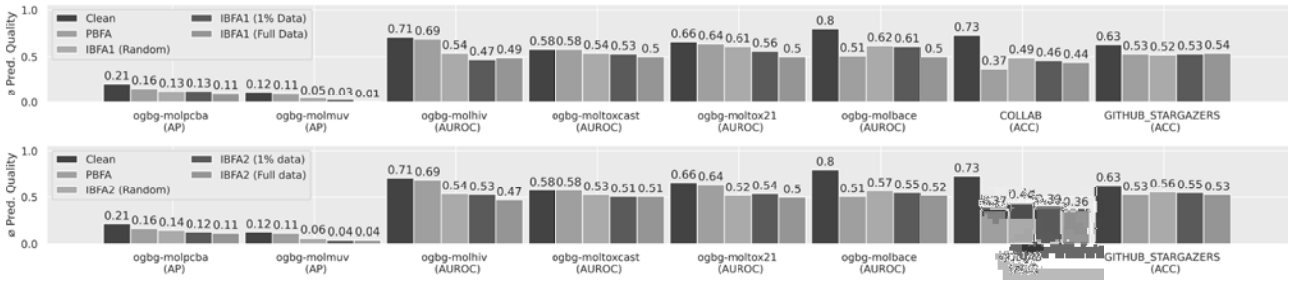


Figure 6: Pre- (clean) and post-attack test quality metrics AP, AUROC or ACC for IBFA with different data selection strategies (random, IBFA selection from 1% random subsets and full training splits) on a 5-layer GIN trained on 6 `ogbg-mol` and 2 TUDataset datasets, averages of 10 runs. IBFA1 in top 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 and CE loss was used for `COLLAB`. Input samples for all evaluated BFA variants were taken from the training splits.

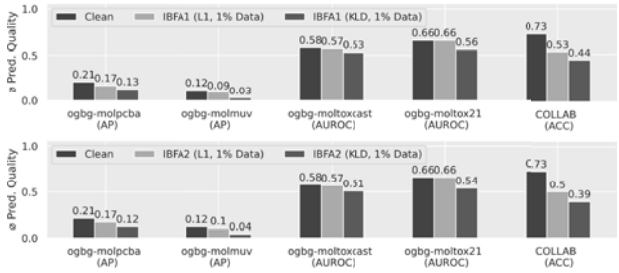


Figure 7: Pre- (clean) and post-attack test quality metrics AP, AUROC or ACC for IBFA with different loss functions (L1 vs. KL loss, IBFA selection from 1% random subset) on a 5-layer GIN trained on 4 `ogbg-mol` datasets TUDataset `COLLAB`, number of bit flips, averages of 10 runs. IBFA1 in top row,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

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`

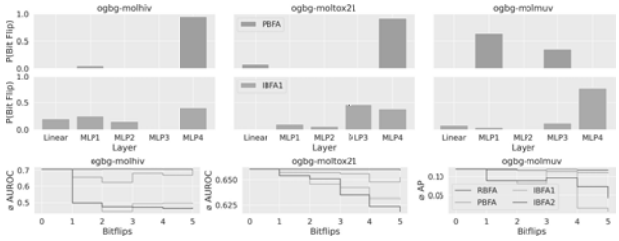


Figure 8: Probability of selecting a layer for a bit flip in a 5-layer GIN trained on 3 small datasets, averaged over 10 runs for PBFA or IBFA1 (top row), progression of degradation of for different BFA variants (bottom row).

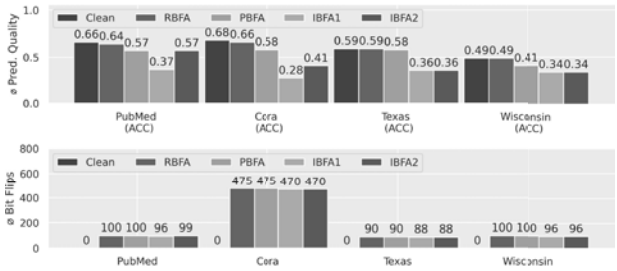


Figure 9: Pre- (clean) and post-attack test quality metric ACC for different BFA variants on a 5-layer GIN trained on four TUDataset node level datasets, number of bit flips, averages of 10 runs/folds.

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.

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.