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 Q

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

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

Here, s denotes the scaling parameter, $\operatorname{clip}(x, a, b) = \min(\max(x, a), b)$ 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:

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

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.

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

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 oversmoothing (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

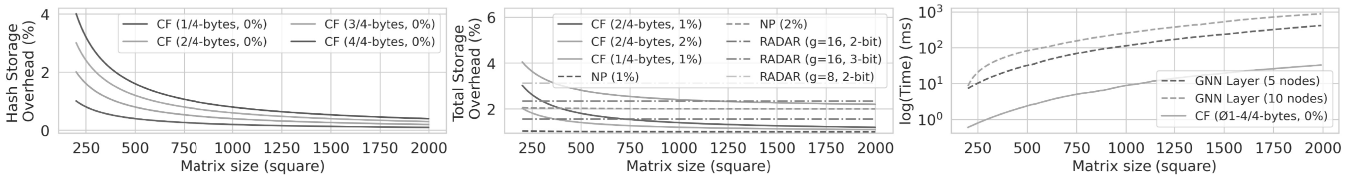

Figure 1: Storage overhead of Crossfire (CF) hashes relative to matrix size (`INT8`), varying cross digest sizes (left). Storage overhead of CF relative to matrix size, varying cross digest sizes and honeypot percentages (%) compared to RADAR and NeuroPots (NP) (center). Average hashing times (milliseconds) 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.

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

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

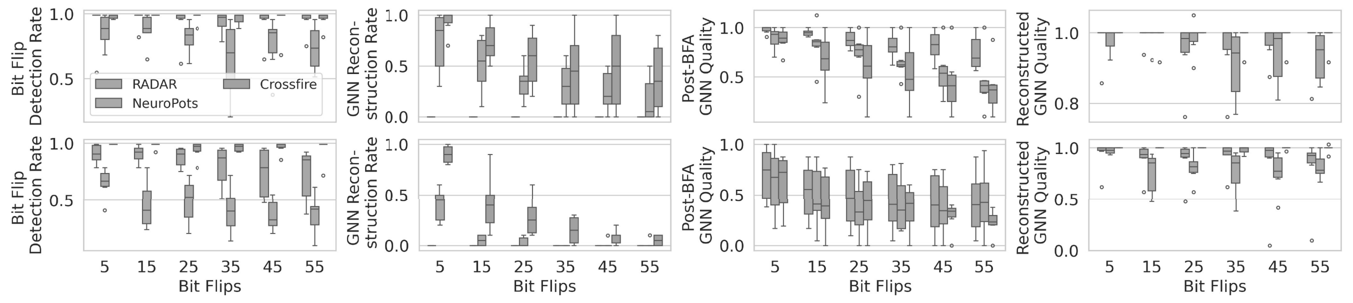

Figure 2: Post-BFA and reconstructed GNN prediction qualities (with pre-attack normalized to 100% to account for different scale quality metrics AP and AUROC), reconstruction, and detection rates for GNNs under PBFA (top row) and IBFA (bottom row) with varying amounts of bit flips, defended by NeuroPots, 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 backpropagation. 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*.