KCES: Training-Free Defense for Robust Graph Neural Networks via Kernel Complexity

Yaning Jia, Shenyang Deng, Chiyu Ma, Yaoqing Yang, Soroush Vosoughi*
Dartmouth College
{yaning.jia.gr, shenyang.deng.gr, soroush.vosoughi}@dartmouth.edu

Abstract

Graph Neural Networks (GNNs) have achieved impressive success across a wide range of graph-based tasks, yet they remain highly vulnerable to small, imperceptible perturbations and adversarial attacks. Although numerous defense methods have been proposed to address these vulnerabilities, many rely on heuristic metrics, overfit to specific attack patterns, and suffer from high computational complexity. In this paper, we propose Kernel Complexity-Based Edge Sanitization (KCES), a training-free, model-agnostic defense framework. KCES leverages Graph Kernel Complexity (GKC), a novel metric derived from the graph's Gram matrix that characterizes GNN generalization via its test error bound. Building on GKC, we define a KC score for each edge, measuring the change in GKC when the edge is removed. Edges with high KC scores, typically introduced by adversarial perturbations, are pruned to mitigate their harmful effects, thereby enhancing GNNs' robustness. KCES can also be seamlessly integrated with existing defense strategies as a plug-and-play module without requiring training. Theoretical analysis and extensive experiments demonstrate that KCES consistently enhances GNN robustness, outperforms state-of-the-art baselines, and amplifies the effectiveness of existing defenses, offering a principled and efficient solution for securing GNNs.

1 Introduction

Graph Neural Networks (GNNs) have emerged as powerful tools for modeling graph-structured data, achieving notable success in diverse domains such as social network analysis [1, 2], recommendation systems [3, 4], and drug discovery [5, 6]. Despite their widespread applicability, GNNs, similar to neural networks operating on Euclidean data [7, 8], are demonstrably vulnerable to adversarial attacks [9, 10, 11, 12]. These attacks involve minor perturbations, such as targeted node manipulations [13] or strategic edge modifications [14], which severely degrades model performance.

To enhance GNN robustness against adversarial attacks, various defense strategies have emerged. Heuristic purification methods like GNN-Jaccard [15] and GNN-SVD [16] refine graph structure by removing suspicious or noisy edges based on similarity or low-rank approximations. Adversarial training approaches, such as RGCN [17], expose the model to perturbed graph during training to learn robust representations. Robust architecture designs like ProGNN [18] and GNNGuard [19] incorporate adaptive graph learning or attention-based edge filtering. However, these methods still exhibit notable limitations: ***(i)*** *Reliance on heuristic metrics:* Many methods (e.g., GNN-Jaccard, GNN-SVD) use simple priors like jaccard similarity or low-rank approximations to identify suspicious edges. These heuristics lack a grounding in model generalization analysis, potentially reducing their effectiveness in identifying edges under adversarial perturbations; ***(ii)*** *Overfitting to specific attacks*: Adversarial training methods like RGCN often defend against known attack patterns (e.g., Nettack [9]), which can limit their robustness to unseen attack strategies; ***(iii)*** *High computational complexity*:

*Corresponding author: soroush.vosoughi@dartmouth.edu

Preprint. Under review.

approaches like ProGNN involve complex iterative optimization procedures, increasing runtime and limiting scalability.

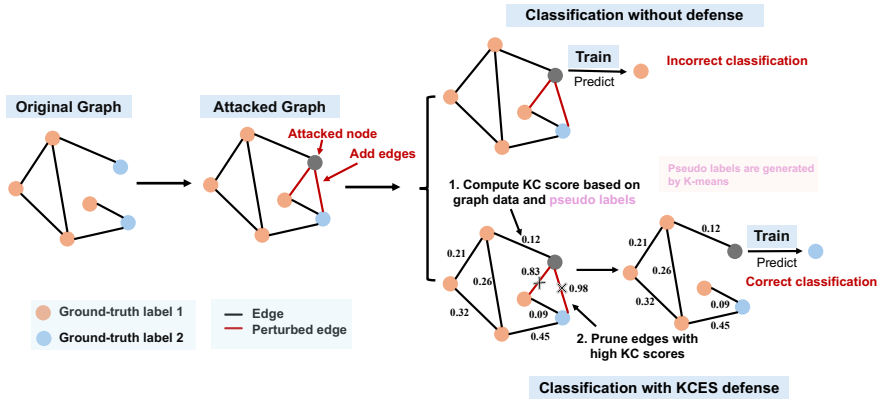

Figure 1: Defense Framework against Adversarial Perturbations with KCES.

Motivated by the above limitations and recent progress in data-independent generalization [20, 21], we introduce **Kernel Complexity-Based Edge Sanitization (KCES)**, which operates in two main steps (see Figure 1). First, drawing on research in kernel models, we introduce **Graph Kernel Complexity (GKC)**, a metric that controls the test error bound of GNN. From GKC, we define the **Graph Kernel Complexity Gap Score (GKCG score)**, or simply the **KC score**, for an edge as the change in GKC upon its removal. This score indicates the corresponding change in the test error bound and serves as an effective signal for identifying adversarial edges, which typically exhibit high KC scores. Second, KCES leverages KC scores, which are computed solely from the graph data and pseudo labels, to perform targeted edge sanitization by pruning edges with high scores. This strategy, validated visually (Figure 1) and experimentally (Sections 6.2 and 6.3), is based on the insight that edges with high KC scores are characteristic of adversarial attacks. By selectively removing these detrimental perturbations while preserving essential benign graph structure, KCES significantly improves GNN robustness.

KCES presents several key advantages over existing defense baselines. It is theoretically grounded under certain assumptions and data-driven, while avoiding reliance on heuristic assumptions about graph properties. This ensures greater flexibility and general applicability across various GNN architectures. KCES quantifies each edge's intrinsic contribution to GNNs' generalization, which measure the probability of the edge under the adversarial perturbations, which is independent of specific attack strategies, helping prevent overfitting to known adversarial patterns and improving robustness against previously unseen attacks. Furthermore, KCES is a training-free, computationally efficient approach, typically needing only a single pre-training computation, thereby bypassing the high costs of iterative optimization methods.

Overall, our contributions are summarized as follows:

- We propose GKC, a novel data-driven, model-agnostic metric defining a key complexity factor in GNN generalization error bounds. From GKC, we derive the KC score to quantify each edge's contribution to test performance on GNNs, effectively identifying adversarially perturbed edges.
- We design KCES, a universal, training-free defense that enhances graph model robustness by selectively pruning likely adversarial edges. This approach is independent of GNN model specifics or attack assumptions.
- Extensive experiments show that KCES outperforms popular defense baselines across diverse datasets and various GNN architectures. Moreover, it can be seamlessly integrated as a plug-and-play component to further boost the robustness of existing defense strategies.

2 Related Works

Attacks and Defenses in GNNs Adversarial attacks in machine learning seek to degrade model performance by introducing subtle input perturbations [22, 23, 24, 25, 26]. When applied to graph-structured data, these attacks compromise structural or semantic integrity through strategies such as

structural perturbations [27, 28, 29, 9, 30], attribute manipulation [11, 13, 31], and malicious node injection [13, 32, 33]. To counter these attacks, various strategies have been developed to enhance the robustness of machine learning models [22, 23, 24, 25, 26, 34, 35]. In the graph domain, several defense methods have been proposed. *Robust architecture design* aims to strengthen GNNs by optimizing their structural components, as demonstrated by GNNGuard [19] and ProGNN [18]. *Graph adversarial training* improves robustness by incorporating adversarial examples during training, as in RGCN [17]. *Graph data purification* mitigates perturbations by denoising or correcting input data, with methods such as GNN-Jaccard [15] and GNN-SVD [16]. However, existing defenses face several challenges, including *insufficient theoretical grounding*, *limited adaptability to diverse attacks*, and *high computational overhead*. These limitations inspire our method's design to overcome them.

Gram Matrix Applications Gram matrices are pivotal for analyzing neural network behavior and optimization. Model-centric applications include investigating how architectures learn target functions [36] and exploring invariance in MLPs [37]. From an optimization standpoint, they are crucial for understanding training dynamics, such as gradient descent in over-parameterized networks [38] and the interplay of learning and stability in two-layer networks [20]. Building on this, training-free data valuation methods employing Gram matrices have quantified individual data point influence in Euclidean data [21]. Such approaches, however, are less explored for Graph Neural Networks (GNNs). Inspired by recent advances [20, 21], we extend this concept to graph-structured data by conceptualizing a two-layer GNN as a kernel model. Then we propose KCES, a training-free and model-agnostic method to enhance GNNs' robustness against adversarial perturbations.

3 Preliminaries

In this section, we summarize the key notations used throughout the paper. Let $\mathbb{R}^d$ denote the d-dimensional Euclidean space, and define $[n] = \{1, 2, \ldots, n\}$. Bold symbols (e.g., $\mathbb{W}$) represent matrices, where $\mathbb{W}_{ij}$ denotes the (i, j)-th entry. If $\mathbb{W}$ is symmetric, then $\lambda_{\min}(\mathbb{W})$ denotes its smallest eigenvalue. Capital letters (e.g., X) denote vectors, and lowercase letters (e.g., c) denote scalars. We use $\|\cdot\|_p$ to denote the p-norm for vectors and the corresponding operator norm for matrices, while $\|\cdot\|_F$ refers to the Frobenius norm. A Gaussian distribution with mean μ and covariance Σ is denoted by $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

3.1 Theoretical Setup for GNNs.

We analyze a two-layer GNN as a kernel model on an undirected graph with N nodes. The graph distribution $\mathcal{D}_G$, defined over $\mathbb{R}^{N\times F} \times \mathbb{R}^N$, generates a single graph $G = (\mathbb{X}, \tilde{\mathbb{A}}, \tilde{\mathbb{D}}, \mathbf{y})$, where each node feature-label pair $(\mathbb{X}_i, y_i) \in \mathbb{R}^F \times \mathbb{R}$ is drawn independently. The adjacency matrix $\tilde{\mathbb{A}} \in \{0, 1\}^{N\times N}$ is fixed, symmetric, and includes self-loops ($\tilde{\mathbb{A}}_{ii} = 1$), and the degree matrix $\tilde{\mathbb{D}}$ is defined by $\tilde{\mathbb{D}}_{ii} = \sum_j \tilde{\mathbb{A}}_{ij}$. A subset of nodes and their corresponding labels from G, denoted as G_{train}, is used for training. The forward propagation of a two-layer GNN for node i is given by:

$$f_{\text{GNN}}(\mathbb{X}_i, \tilde{\mathbb{A}}, \tilde{\mathbb{D}}) = \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \sigma\left(W_r^\top \left(\tilde{\mathbb{D}}^{-\frac{1}{2}} \tilde{\mathbb{A}} \tilde{\mathbb{D}}^{-\frac{1}{2}} \mathbb{X}\right)_i\right), \tag{1}$$

Here, m denotes the number of hidden units; $\mathbb{W} \in \mathbb{R}^{F\times m}$ is the first-layer weight matrix with columns $W_r \in \mathbb{R}^F$; $\mathbf{a} = (a_1, \ldots, a_m)^\top \in \mathbb{R}^m$ is the second-layer weight vector with scalar entries a_r; $\sigma(\cdot)$ is the activation function (e.g., ReLU).

For the GNN $f_{\text{GNN}}(\mathbb{X}_i, \tilde{\mathbb{A}}, \tilde{\mathbb{D}})$, we define the training error (empirical risk) on G_{train} as:

$$L(\mathbb{W}) = \frac{1}{2} \sum_{i=1}^{N} \left(y_i - f_{\text{GNN}}(\mathbb{X}_i, \tilde{\mathbb{A}}, \tilde{\mathbb{D}})_i\right)^2, \tag{2}$$

Here, $\mathbb{X}_i$ and y_i are sampled from G_{train}. The corresponding test error (expected risk) is defined as the expectation of the training error over the graph distribution $\mathcal{D}_G$, denoted by:

$$L_{\mathcal{D}_G}(\mathbb{W}) = \mathbb{E}_{G\sim\mathcal{D}_G}\left[L(\mathbb{W})\right]. \tag{3}$$

3.2 Graph Kernel Gram Matrix

We introduce the **Graph Kernel Gram Matrix** (hereafter, the **Gram matrix**), which captures pairwise node relationships based solely on graph structure and node features. This matrix builds upon classical kernel methods [39, 20, 37, 40], where Gram matrices are widely used to connect model complexity with generalization performance. In our specific GNN setting, *we integrate the graph aggregation operation directly into this kernel*. Consequently, for a graph $G = (\mathbb{X}, \tilde{\mathbb{A}}, \tilde{\mathbb{D}}, \mathbf{y})$, given the normalized aggregated feature matrix $\tilde{\mathbb{X}} = \tilde{\mathbb{D}}^{-\frac{1}{2}}\tilde{\mathbb{A}}\tilde{\mathbb{D}}^{-\frac{1}{2}}\mathbb{X} \in \mathbb{R}^{N\times F}$, we define the Gram matrix $\mathbb{H}^{\infty} = \left[\mathbb{H}^{\infty}_{ij}\right]_{i,j=1}^{N} \in \mathbb{R}^{N\times N}$, where each entry $\mathbb{H}^{\infty}_{ij}$ is given by:

$$\mathbb{H}^{\infty}_{ij} = \frac{\tilde{\mathbb{X}}_i^{\top}\tilde{\mathbb{X}}_j\left(\pi - \arccos\left(\tilde{\mathbb{X}}_i^{\top}\tilde{\mathbb{X}}_j\right)\right)}{2\pi}, \tag{4}$$

Here, $\tilde{\mathbb{X}}_i \in \mathbb{R}^{1\times F}$ and $\tilde{\mathbb{X}}_j \in \mathbb{R}^{1\times F}$ represent the i-th and j-th rows of the matrix $\tilde{\mathbb{X}}$, respectively. The Gram matrix $\mathbb{H}^{\infty}$, computed from these aggregated features, captures the pairwise interactions between nodes. This process effectively establishes a kernel-induced feature space, which forms the basis for computing the Graph Kernel Complexity presented in the subsequent section.

3.3 Graph Kernel Complexity

Based on the above Gram matrix, we define the **Graph Kernel Complexity (GKC)** to qualify the test error bound of GNNs, reflecting their generalization capacity. Formally, given the Gram matrix $\mathbb{H}^{\infty} \in \mathbb{R}^{N\times N}$ and the label vector $\mathbf{y} \in \mathbb{R}^{N}$, the GKC is defined as:

$$\mathrm{GKC}(\mathbb{H}^{\infty}) = \frac{2\mathbf{y}^{\top}\left(\mathbb{H}^{\infty}\right)^{-1}\mathbf{y}}{N}. \tag{5}$$

Notably, $\mathbf{y}$ *denotes pseudo labels generated by K-means [41] in practice, rather than the ground-truth node labels.* Therefore, GKC is a *model-independent* complexity metric, determined solely by intrinsic data properties. A lower GKC score reflects well-aligned and internally consistent data, which simplifies the learning process and facilitates generalization in GNNs. Theorem 4.2 formalizes this relationship by showing that smaller GKC values correspond to lower test error, thereby indicating stronger generalization.

4 GKC-based Generalization Analysis

This section employs the Gram matrix and GKC to establish generalization bounds for the two-layer GNN (Section 3.1). Theorems for training and test errors are presented informally for clarity, with formal statements and proofs in the Appendix.

First, Theorem 4.1 characterizes the training error dynamics as following:

Theorem 4.1 (**Training error dynamics**)**.** Under Assumption E.2 in Appendix E, after t gradient descent updates with step size η, the training error satisfies

$$L(\mathbb{W}_t) = \left\|\left(\mathbb{I} - \eta\mathbb{H}^{\infty}\right)^{t}\mathbf{y}\right\|_2^2 + \varepsilon, \tag{6}$$

where $\mathbb{H}^{\infty}$ denotes the Gram matrix, m is the hidden layer width, and $\varepsilon = \tilde{O}(m^{-1/2})$ represents an error term dependent on t and m. The formula shows that the training error decreases with the number of training steps, and the rate of decrease is governed by the Gram matrix. A formal version of the result is provided in Appendix E.2.1 (Theorem E.4).

Building on training error analysis, Theorem 4.2 establishes a generalization bound via GKC:

Theorem 4.2 (**Test error bound**)**.** Under Assumption E.2 in Appendix E, for sufficiently large hidden layer width m and iteration count t, the following holds with probability at least $1-\delta$:

$$L_{\mathcal{D}_G}(\mathbb{W}_t) \;\le\; \sqrt{\mathrm{GKC}(\mathbb{H}^{\infty})} \;+\; O\left(\sqrt{\frac{\log\frac{N}{\lambda_0\delta}}{N}}\right), \tag{7}$$

Here, GKC($\mathbb{H}^\infty$) denotes the Graph Kernel Complexity (Section 3.3), $\delta \in (0,1)$ is the confidence level, N is the number of nodes, and λ_0 is a lower bound on $\lambda_{\min}(\mathbb{H}^\infty)$ (Lemma E.3). The generalization bound on the GNN test error, as derived from the formula, is primarily influenced by the data-dependent GKC term. A smaller GKC leads to a tighter bound, indicating improved generalization under standard GNN training regimes. The formal statement and proof are presented in Appendix E.2.1 (Theorem E.5).

5 Kernel Complexity-Based Edge Sanitization for Robustness

Motivated by the established connection between GKC and the test error of GNNs, we propose **Kernel Complexity-Based Edge Sanitization** (**KCES**), a training-free and model-agnostic defense framework that evaluates the importance of graph edges and strategically prunes them to improve GNN robustness against adversarial perturbations. This framework comprises two key components: ***(i) Edge KC Score Estimation*** and ***(ii) KC-Based Edge Sanitization***. The first component assigns a kernel-based score to each edge, quantifying its structural importance. The second component utilizes these scores to selectively remove potentially harmful edges, thereby mitigating the impact of adversarial attacks.

5.1 Edge KC Score Estimation

To quantify the importance of each edge, we introduce the **Graph Kernel Complexity Gap Score**, hereafter referred to as the **KC score**. Specifically, for an edge e_{ij} connecting the i-th and j-th nodes in the graph, the KC score is defined as:

$$KC(i,j) = \left| \text{GKC}\left(\mathbb{H}^\infty\right) - \text{GKC}\left(\mathbb{H}^\infty_{-(i,j)}\right) \right|, \tag{8}$$

Here, $\mathbb{H}^\infty$ denotes the Gram matrix of the original graph, and $\mathbb{H}^\infty_{-(i,j)}$ corresponds to the Gram matrix of the graph $G_{-(i,j)}$, obtained by removing edge e_{ij}. Both matrices are computed using pseudo labels $\mathbf{y}$ generated by *K-means*; the details of generating $\mathbf{y}$ refers to Appendix B. The KC score $KC(i,j)$ quantifies the change in GKC resulting from the removal of edge e_{ij}. This score governs the upper bound of the interval within which the test error may vary, as stated in Corollary 5.1 (For a formal version and proof of the corollary, see Appendix E.2.2 and E.4.3).

Corollary 5.1 (Edge-specific test error bound)**.** Let the graph $G_{-(i,j)}$ be obtained by deleting edge e_{ij}, and let $\mathbb{W}_t$ denote the GNN parameters after t gradient descent updates on the modified graph. Under the same assumptions as Theorem 4.2, and for sufficiently large hidden width m and iteration count t, the following holds with probability at least $1-\delta$:

$$L_{\mathcal{D}_{G_{-(i,j)}}}\left(\mathbb{W}_t\right) \leq \sqrt{\text{GKC}\left(\mathbb{H}^\infty\right)} + \sqrt{KC(i,j)} + O\left(\sqrt{\frac{\log \frac{N}{\lambda_0 \delta}}{N}}\right), \tag{9}$$

where N denotes the number of nodes; λ_0 is the lower bound on $\lambda_{\min}(\mathbb{H}^\infty)$, as specified in Assumption E.2; and $\mathcal{D}_{G_{-(i,j)}}$ represents the data distribution for graph $G_{-(i,j)}$, which are generated by taking graphs from $\mathcal{D}_G$ and removing edge e_{ij}. Corollary 5.1 establishes a theoretical link between the KC score of an edge e_{ij} and the GNN's expected test error on the graph $G_{-(i,j)}$ (i.e., after removing e_{ij}), thereby quantifying the edge's contribution to the overall GKC. Adversarially perturbed edges, being theoretically detrimental to GNN generalization, are expected to exhibit higher KC scores. This is empirically supported by the results in Section 6.2, which shows attacks increase the prevalence of such high-impact edges. Thus, the KC score serves as an effective indicator of these adversarial structural perturbations, guiding the strategic removal of detrimental edges to enhance GNN robustness.

5.2 KC-based Edge Sanitization

Building on estimated KC scores, we can use them to sanitize graphs. Since edges with high scores tend to negatively impact model performance, we can remove edges with higher KC scores to ensure GNN performance, enhancing its robustness. Specific **KCES** refer to the following Algorithm 1.

Algorithm 1 KCES: Kernel Complexity-Based Edge Sanitization

Input: Graph $G = (V, E)$; features $\mathbb{X}$; pruning ratio $\alpha \in [0, 1]$
Output: Sanitized graph $G' = (V, E')$

$\hat{\mathbf{y}} \leftarrow \text{K-means}(\tilde{\mathbb{A}}, \mathbb{X})$ *// Generate pseudo labels for nodes via K-means*
$\mathbb{H}^{\infty} \leftarrow \text{GKGM}(G, \mathbb{X})$ *// Compute original Gram matrix, see Eq. 4*
$\text{GKC}(\mathbb{H}^{\infty}) \leftarrow \frac{2\hat{\mathbf{y}}^{\top}(\mathbb{H}^{\infty})^{-1}\hat{\mathbf{y}}}{N}$ *// Compute original GKC, see Eq. 5*
$\text{KC} \leftarrow [\,]$ *// Initialize a KC list for edge scores*
for each edge $e_{ij} \in E$ **do**
  $G_{-(i,j)} \leftarrow G \setminus \{e_{ij}\}$ *// Create a modified graph excluding edge e_{ij}*
  $\mathbb{H}^{\infty}_{-(i,j)} \leftarrow \text{GKGM}(G_{-(i,j)}, \mathbb{X})$ *// Compute Gram matrix for the modified graph, see Eq. 4*
  $\text{GKC}(\mathbb{H}^{\infty}_{-(i,j)}) \leftarrow \frac{2\hat{\mathbf{y}}^{\top}(\mathbb{H}^{\infty}_{-(i,j)})^{-1}\hat{\mathbf{y}}}{N}$ *// Compute GKC for the modified graph, see Eq. 5*
  $\text{KC}[e_{ij}] \leftarrow \left|\text{GKC}(\mathbb{H}^{\infty} - \text{GKC}(\mathbb{H}^{\infty}_{-(i,j)})\right|$ *// Compute KC score for e_{ij}, see Eq. 8*
end for
$(e_{(1)}, e_{(2)}, \ldots, e_{(|E|)}) \leftarrow \text{SortEdgesByScore}(\text{KC}, \text{descending})$ *// Sort edges by KC score*
$k \leftarrow \lceil \alpha \cdot |E| \rceil$ *// Number of edges to remove*
$E' \leftarrow E \setminus \{e_{(1)}, \ldots, e_{(k)}\}$ *// Prune the top-k KC score edges*
Return $G' = (V, E')$ *// Return Graph with pruned edge set*

6 Experiments

This section details four comprehensive experiments designed to validate our theoretical analyses and assess the KCES method's effectiveness against adversarial attacks. **Experiment 1** examines KC score distributions in clean and adversarial graphs, explaining the effect of adversarial perturbations on KC score. **Experiment 2** shows connection between pruning strategies and model generalization, empirically demonstrating edge KC score is highly linked with GNN performance. **Experiment 3** assesses KCES's robustness across diverse adversarial settings, demonstrating state-of-the-art results. Finally, **Experiment 4** showcases KCES's plug-and-play compatibility with existing defense baselines, highlighting its capacity to further improve their robustness.

6.1 Overall Setup

Dataset We evaluate the robustness and scalability of the proposed KCES method using five benchmark datasets spaning three distinct scales: **(1)** small-scale graphs (fewer than 10,000 edges): *Cora*[42], *Citeseer* [42], and *Polblogs[43]*; **(2)** medium-scale graph (10,000-100,000 edges): *Pubmed* [44]; and **(3)** large-scale graph (exceeding 100,000 edges): *Flickr*[45]. Detailed dataset statistics are provided in Appendix A.

Attack strategies We utilize five representative attack methods to evaluate robustness, categorized into three types. *These strategies involve perturbing the graph data before GNN training, with defense methods subsequently applied to mitigate their impact.* The categories are: ***(i) Non-targeted Attack***, which degrade overall GNN performance by perturbing the entire graph structure, including *Metattack* [30], *MINMAX* [27], and *DICE* [29]; ***(ii) Targeted Attack***, which focus on misleading GNN predictions for specific nodes, for which we employ the widely-used *Nettack* [9]; and ***(iii) Random Attack***, which simulate noise by randomly adding or removing edges at, termed *Random*.

Baseline methods We evaluate the effectiveness of KCES by comparing it against several state-of-the-art defense methods from two complementary perspectives. First, we consider a set of widely adopted and robust GNN architectures, including GCN [27], GAT [46], and RGCN [17]. Second, we benchmark KCES against established defense strategies specifically designed to improve the robustness of GNN in adversarial settings. These include ProGNN [18], GNN-Jaccard [15], GNN-SVD [16], and GNNGuard [19], all of which represent mainstream approaches for defending against both non-targeted and targeted attacks.

Implementation details We use test accuracy for node classification as the primary evaluation metric in the following experiments. Further implementation details are provided in Appendix C. Code is available at: `https://anonymous.4open.science/r/KCScore-FB15/README.md`.

6.2 Impact of Adversarial Perturbations on Kernel Complexity Gap Scores

This experiment investigates the ***relationship between KC score distribution and adversarial perturbations***. Based on Corollary 5.1, adversarially perturbed edges are usually detrimental to GNN generalization, which may increase GKC theoretically, resulting in a larger test error bound (expressed as large KC scores). To verify this, we compare KC score distributions across three settings: . To verify this, we compare KC score distributions across three settings: **(1)** the original clean graph, **(2)** an adversarially perturbed graph (using *Metattack*), and **(3)** the perturbed graph after pruning its high-KC score edges. For a fair comparison, an equal number of edges are randomly sampled from each graph type, and their KC scores are normalized to the $[0, 1]$ range. Kernel Density Estimation (KDE) [47] is used to visualize these smoothed distributions. Experiments are conducted on *Cora* and *Pubmed* datasets (representing varying graph scales) using a GCN. We sample 1,000 edges from *Cora* and 10,000 from *Pubmed*, proportional to their sizes. Pruning ratios are set to 0.25 for *Cora* and 0.75 for *Pubmed*, based on findings in Section 6.3.

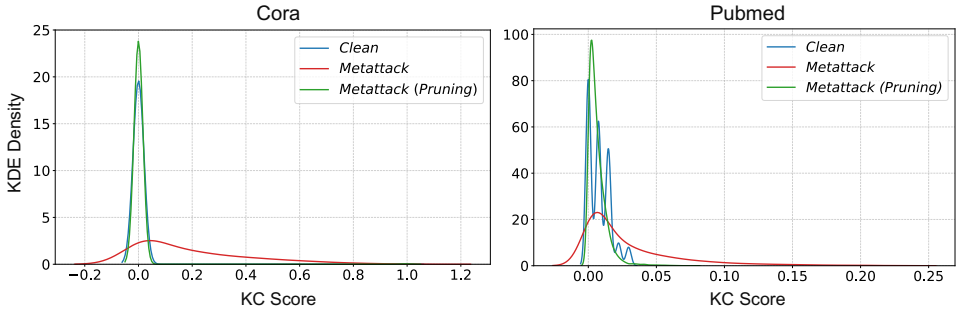

Figure 2: Visualization of KC Score Distributions Across *Clean*, *Metattack*, and *Pruned Graphs*.

The results, depicted in Figure 2, demonstrate that adversarial perturbations markedly increase edge KC scores, thereby empirically supporting Corollary 5.1. Specifically, the KC score distribution for clean graphs is highly concentrated around zero, reflecting a low prevalence of edges with high KC scores. In contrast, *Metattack* induces a significant shift in the distribution towards higher KC values and introduces a long tail. This suggests an increase in structurally redundant or uninformative edges, which adversely affect GNN performance as edges with high KC scores contribute less to effective learning. The proposed pruning strategy effectively counteracts this shift by removing these high-KC score edges, thereby restoring the distribution to a form that closely resembles that of the clean graph. These findings align with our theoretical analysis in Section 4, which posits that edges with high KC scores are significant contributors to graph complexity and thus detrimental. The subsequent section will further explore the impact of edges with high and low KC scores on GNN performance.

6.3 Impact of Edge Sanitization on Model Performance

Section 6.2 and Corollary 5.1 establish a link between each edge's influence on GNN performance and its corresponding test error, as quantified by the KC score. Edges with lower KC scores (Low-KC), such as unperturbed ones, are typically associated with lower test error and improved GNN performance. In contrast, edges with higher KC scores (High-KC), often resulting from adversarial perturbations, tend to increase test error and degrade performance. To empirically validate this insight, we prune edges based on their KC scores and assess the impact on model performance. Three distinct pruning strategies are evaluated: ***(i) High-KC Pruning***, removing edges with the highest KC scores; ***(ii) Low-KC Pruning***, removing edges with the lowest KC scores; and ***(iii) Random Pruning***, removing edges randomly as a baseline. Each strategy is tested with pruning ratios ranging from 0.05 to 0.95, in 0.05 increments. The primary evaluations utilize adversarially perturbed versions of the *Cora* and *Pubmed* datasets (attacked via *Metattack*), with a GCN architecture employed consistently. Results from corresponding experiments on clean graphs are detailed in Appendix D.1.

The result presented in Figure 6.3 highlights the significant positive contribution of Low-KC edges to GNN performance, in contrast to the demonstrably negative impact of High-KC edges. This finding, reinforced by analogous results on clean graphs (Appendix D.1), validates High-KC Pruning as an effective method for removing detrimental edges in adversarial scenarios. It thus represents

a robust strategy for improving GNN generalization and bolstering robustness against adversarial perturbations. Specifically, under *Metattack*, High-KC Pruning consistently sustains or improves GNN performance. In stark contrast, Low-KC Pruning generally leads to a decline in accuracy, often performing worse than Random Pruning. Notably, under adversarial conditions, both Low-KC and Random Pruning can yield performance improvements at high pruning ratios (beyond 0.5). This phenomenon is likely attributable to the extensive removal of adversarial edges, where the benefit of eliminating numerous detrimental perturbations outweighs the cost of losing some benign edges. High-KC Pruning, however, does not exhibit this phenomenon, suggesting its superior precision in targeting harmful edges while preserving essential ones. Overall, the efficacy of these pruning strategies follows the order: ***High-KC Pruning > Random Pruning > Low-KC Pruning***. These empirical results align well with our theoretical analyses in Section 4.

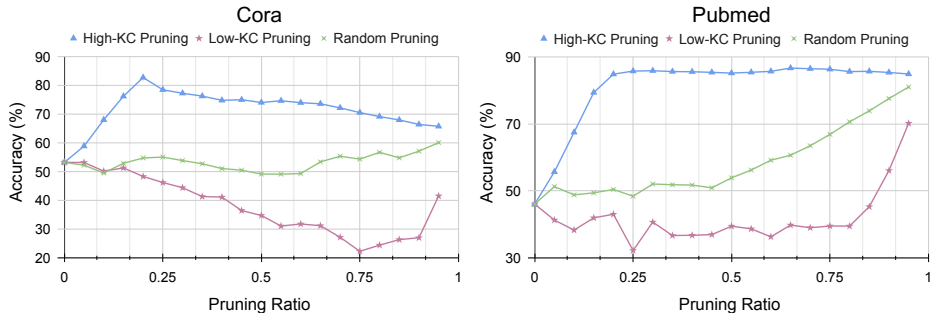

Figure 3: Comparing Edge Pruning Strategies via KC Scores in Adversarial Settings.

6.4 Defense against adversarial attacks

In this section, we evaluate that KCES can improve GNNs' robustness against various adversarial attacks. The experimental setup, including datasets, attack methods, and defense baselines, follows the configuration described in Section 6.1. For non-targeted and random attacks, we allocate 25% of the graph's edges as the perturbation budget. For the targeted attack (*Nettack*), we select nodes with degrees greater than 10 and perturb their connected edges. We implement KCES on GCN and compare its robustness against other baseline methods. All results are reported as percentages, and N/A indicates that the method is not applicable to the corresponding attack setting.

Table 1 reports the defense performance of various methods across five datasets under random, untargeted, and targeted attacks. Overall, our proposed method, KCES, consistently outperforms baselines and achieves state-of-the-art robustness across all attack settings. Notably, KCES enables the model to match, and occasionally exceed its performance on clean setting, even under adversarial perturbations. On the *Flickr* dataset, we observe that several baselines perform better under adversarial conditions than on clean data. This is likely due to the presence of numerous noisy or irrelevant edges in the large-scale graph, which hinder model performance. This pattern is consistent with prior findings [48, 49], suggesting that pruning-based defense methods, including ours, can reduce the impact of such noise, effectively acting as regularization and enhancing both robustness and generalization. Overall, these results confirm the effectiveness of KCES over baselines.

6.5 Effect of KCES as a Plug-and-Play Module for Enhancing Existing Defenses

KCES is a training-free and data-independent method, making it highly adaptable as a plug-and-play module to enhance the robustness of existing models. To demonstrate its versatility and effectiveness, we apply KCES to several baseline methods listed in Table 1, including robust GNN architectures (GAT and RGCN) as well as defense algorithms (ProGNN, GNN-SVD, GNN-Jaccard, and GNNGuard).

The results, presented in Figure 4, demonstrate that integrating KCES consistently improves the performance of all baseline methods across diverse adversarial attacks, underscoring its effectiveness as a plug-and-play module. Notably, KCES substantially enhances the robustness of both GNN architectures and defense algorithms, with all KCES-integrated baselines achieving over 70% accuracy on *Cora* and over 80% on *Pubmed* under a wide range of adversarial attacks. These findings indicate that KCES does not conflict with existing defense methods and can be seamlessly integrated to further strengthen their robustness against attacks.

Table 1: Defense performance (Accuracy ± Std) under targeted and untargeted adversarial attacks.

Dataset	Attack	GCN	GAT	RGCN	ProGNN	GNN-SVD	GNN-Jaccard	GNNGuard	KCES (Ours)
Polblogs (*small*)	*Clean*	95.71 ± 0.79	95.40 ± 0.41	95.29 ± 0.52	95.60 ± 0.46	94.68 ± 0.88	N/A	N/A	**96.01 ± 0.18**
	Random	81.29 ± 0.61	85.01 ± 0.02	82.23 ± 0.76	86.50 ± 0.12	88.34 ± 0.83	N/A	N/A	**89.46 ± 0.77**
	Nettack	92.59 ± 0.73	85.79 ± 0.23	93.15 ± 0.70	95.20 ± 0.59	95.37 ± 0.57	N/A	N/A	**96.48 ± 0.82**
	DICE	71.47 ± 0.98	77.10 ± 0.87	71.16 ± 0.36	74.74 ± 0.18	76.48 ± 0.73	N/A	N/A	**93.04 ± 0.06**
	MINMAX	70.14 ± 0.54	68.04 ± 0.89	82.10 ± 0.42	60.73 ± 0.37	62.47 ± 0.72	N/A	N/A	**94.17 ± 0.47**
	Metattack	63.09 ± 0.12	63.60 ± 0.20	62.67 ± 0.13	65.23 ± 0.24	81.28 ± 0.58	N/A	N/A	**82.72 ± 0.32**
Cora (*small*)	*Clean*	83.65 ± 0.57	83.90 ± 0.15	82.89 ± 0.99	**85.16 ± 0.77**	78.16 ± 0.07	82.69 ± 0.74	78.87 ± 0.77	84.04 ± 0.42
	Random	77.57 ± 0.38	79.23 ± 0.75	74.55 ± 0.13	79.58 ± 0.32	78.87 ± 0.36	77.36 ± 0.93	77.11 ± 0.57	**79.67 ± 0.69**
	Nettack	57.83 ± 0.78	58.23 ± 0.08	59.04 ± 0.51	69.67 ± 0.68	74.70 ± 1.21	76.69 ± 0.64	62.65 ± 0.03	**82.24 ± 0.43**
	DICE	76.16 ± 0.69	77.06 ± 0.31	73.59 ± 0.91	75.95 ± 0.40	72.68 ± 0.72	77.11 ± 0.22	75.50 ± 0.26	**82.59 ± 0.09**
	MINMAX	60.66 ± 0.23	61.26 ± 0.73	59.80 ± 0.75	64.43 ± 0.23	59.45 ± 0.10	72.23 ± 0.69	70.57 ± 0.95	**78.42 ± 0.47**
	Metattack	53.12 ± 0.83	58.35 ± 0.22	51.35 ± 0.72	63.37 ± 0.16	61.92 ± 0.28	75.28 ± 0.19	70.77 ± 0.97	**82.99 ± 0.08**
Citeseer (*small*)	*Clean*	72.51 ± 0.61	72.69 ± 0.55	71.97 ± 0.60	71.74 ± 0.59	69.60 ± 0.56	72.98 ± 0.06	71.03 ± 0.19	**73.02 ± 0.93**
	Random	70.38 ± 0.83	69.31 ± 0.95	67.06 ± 0.26	72.36 ± 0.19	67.59 ± 0.15	71.21 ± 0.03	72.73 ± 0.60	**72.81 ± 0.62**
	Nettack	52.38 ± 0.94	59.19 ± 0.46	49.21 ± 0.97	72.23 ± 1.04	74.60 ± 0.51	72.14 ± 0.69	72.95 ± 0.58	**76.14 ± 0.60**
	DICE	67.71 ± 0.72	66.60 ± 0.43	66.17 ± 0.85	72.15 ± 0.61	67.35 ± 0.03	71.14 ± 0.19	69.01 ± 0.34	**72.45 ± 0.84**
	MINMAX	66.29 ± 0.45	67.54 ± 0.43	61.02 ± 0.44	69.90 ± 0.80	64.57 ± 0.86	71.20 ± 0.02	68.60 ± 0.86	**72.80 ± 0.36**
	Metattack	57.64 ± 0.59	61.20 ± 0.66	56.81 ± 0.75	66.33 ± 0.46	66.29 ± 0.39	70.14 ± 0.53	64.75 ± 0.60	**71.86 ± 0.87**
Pubmed (*medium*)	*Clean*	85.72 ± 0.05	84.89 ± 0.84	84.72 ± 0.06	85.19 ± 0.43	84.53 ± 0.86	86.19 ± 0.97	84.49 ± 0.79	**86.17 ± 0.52**
	Random	84.11 ± 0.28	81.02 ± 0.72	83.75 ± 0.10	84.28 ± 0.41	82.61 ± 0.63	84.27 ± 0.64	83.87 ± 0.03	**85.83 ± 0.82**
	Nettack	66.67 ± 0.36	76.73 ± 0.88	72.58 ± 0.09	72.60 ± 0.14	80.10 ± 0.45	85.48 ± 0.12	83.33 ± 0.65	**86.24 ± 0.09**
	DICE	81.68 ± 0.46	76.93 ± 0.19	81.44 ± 0.56	80.73 ± 0.30	80.39 ± 0.12	82.93 ± 0.70	82.28 ± 0.70	**85.74 ± 0.39**
	MINMAX	55.67 ± 0.46	60.01 ± 0.97	54.64 ± 0.24	69.29 ± 0.37	80.50 ± 0.06	84.51 ± 0.83	81.69 ± 0.57	**85.64 ± 0.97**
	Metattack	46.08 ± 0.39	49.72 ± 0.37	45.99 ± 0.92	72.08 ± 0.51	82.75 ± 0.25	84.22 ± 0.39	83.37 ± 0.89	**86.45 ± 0.45**
Flickr (*large*)	*Clean*	56.24 ± 0.81	47.83 ± 0.25	39.62 ± 0.73	54.68 ± 0.68	60.46 ± 0.55	74.04 ± 0.38	74.31 ± 0.74	**76.25 ± 0.93**
	Random	62.82 ± 0.62	60.80 ± 0.42	62.44 ± 0.16	65.43 ± 0.90	76.54 ± 0.41	74.83 ± 0.96	74.85 ± 0.55	**76.74 ± 0.65**
	Nettack	38.82 ± 0.55	43.12 ± 0.39	59.87 ± 0.36	70.31 ± 0.74	58.70 ± 0.45	72.58 ± 0.48	**74.51 ± 0.33**	73.47 ± 0.97
	DICE	51.71 ± 0.32	48.11 ± 0.11	47.34 ± 0.39	71.23 ± 0.17	73.38 ± 0.93	73.35 ± 0.02	73.59 ± 0.24	**73.69 ± 0.67**
	MINMAX	14.57 ± 0.97	11.71 ± 0.26	27.13 ± 0.02	19.68 ± 0.58	39.17 ± 0.78	74.82 ± 0.33	75.04 ± 0.08	**76.86 ± 0.90**
	Metattack	36.93 ± 0.86	37.29 ± 0.06	31.72 ± 0.21	65.05 ± 0.82	59.08 ± 0.36	74.85 ± 0.95	75.05 ± 0.75	**76.63 ± 0.82**

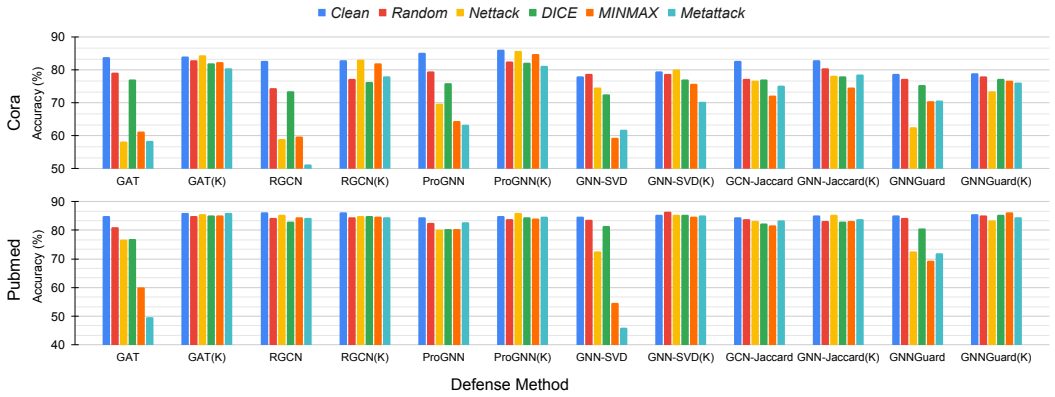

Figure 4: Robustness Enhancement Against Adversarial Attacks via **KCES** Integration.

7 Conclusion and Limitation

In this work, we proposed KCES, a training-free and model-agnostic framework for defending GNNs against adversarial attacks. At its core is the GKC, a novel metric derived from the graph's Gram matrix to quantify generalization. Building on GKC, we introduced the KC score, which measures the impact of each edge on generalization by evaluating the change in GKC after edge removal. Our theoretical analysis established a direct connection between KC scores and test error, enabling principled edge sanitization. Experiments across diverse datasets and attack settings showed that KCES consistently outperforms baselines and can serve as a plug-and-play component for improving existing defenses. Despite its demonstrated effectiveness, KCES possesses certain limitations. Firstly, its primary focus is on structural perturbations, meaning it does not inherently address adversarial attacks targeting node features. Secondly, as an edge sanitization technique, KCES is not directly applicable to graph-based tasks that are intrinsically edge-centric. Thirdly, while its implementation is model-agnostic, the theoretical underpinnings of KCES rely on assumptions specific to GNNs, which may limit its direct generalization to architectures outside the GNN family. Overcoming these limitations could lead to more robust and versatile iterations of KCES, thereby further advancing GNN security in complex, real-world applications.

References

[1] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[2] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016.

[3] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018.

[4] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020.

[5] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, 2017.

[6] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, et al. Analyzing learned molecular representations for property prediction. *Journal of Chemical Information and Modeling*, 2019.

[7] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, 2017.

[8] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, 2020.

[9] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018.

[10] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[11] Aleksandar Bojchevski and Stephan Günnemann. Adversarial attacks on node embeddings via graph poisoning. In *International conference on machine learning*, 2019.

[12] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International conference on machine learning*, 2018.

[13] Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In *Proceedings of the Web Conference 2020*, 2020.

[14] Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. Robustness of graph neural networks at scale. *Advances in Neural Information Processing Systems*, 2021.

[15] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples for graph data: deep insights into attack and defense. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019.

[16] Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th international conference on web search and data mining*, 2020.

[17] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019.

[18] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020.

[19] Xiang Zhang and Marinka Zitnik. Gnnguard: Defending graph neural networks against adversarial attacks. In *Advances in neural information processing systems*, 2020.

[20] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International conference on machine learning*, 2019.

[21] Ki Nohyun, Hoyong Choi, and Hye Won Chung. Data valuation without training of a model. In *The Eleventh International Conference on Learning Representations*, 2022.

[22] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[23] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[24] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, 2016.

[25] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018.

[26] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[27] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui Wei Weng, Mingyi Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective. In *International Joint Conference on Artificial Intelligence*, 2019.

[28] Jinyin Chen, Yangyang Wu, Xuanheng Xu, Yixian Chen, Haibin Zheng, and Qi Xuan. Fast gradient attack on network embedding. *arXiv preprint arXiv:1809.02797*, 2018.

[29] Marcin Waniek, Tomasz P Michalak, Michael J Wooldridge, and Talal Rahwan. Hiding individuals and communities in a social network. *Nature Human Behaviour*, 2018.

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

[31] Fuli Feng, Xiangnan He, Jie Tang, and Tat-Seng Chua. Graph adversarial training: Dynamically regularizing based on graph structure. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[32] Xu Zou, Qinkai Zheng, Yuxiao Dong, Xinyu Guan, Evgeny Kharlamov, Jialiang Lu, and Jie Tang. Tdgia: Effective injection attacks on graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021.

[33] Shuchang Tao, Qi Cao, Huawei Shen, Junjie Huang, Yunfan Wu, and Xueqi Cheng. Single node injection attack against graph neural networks. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021.

[34] Yaning Jia, Dongmian Zou, Hongfei Wang, and Hai Jin. Enhancing node-level adversarial defenses by lipschitz regularization of graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023.

[35] Yaning Jia, Chunhui Zhang, and Soroush Vosoughi. Aligning relational learning with lipschitz fairness. In *The Twelfth International Conference on Learning Representations*, 2024.

[36] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[37] Russell Tsuchida, Fred Roosta, and Marcus Gallagher. Invariance of weight distributions in rectified mlps. In *International Conference on Machine Learning*, 2018.

[38] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International conference on machine learning*, 2019.

[39] Bo Xie, Yingyu Liang, and Le Song. Diverse neural network learns true target functions. In *Artificial Intelligence and Statistics*, 2017.

[40] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

[41] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, 1967.

[42] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 2008.

[43] Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, 2005.

[44] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, 2016.

[45] Huan Liu, John Salerno, and Michael J Young. *Social computing and behavioral modeling*. Springer Science & Business Media, 2009.

[46] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[47] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 1962.

[48] Enyan Dai, Wei Jin, Hui Liu, and Suhang Wang. Towards robust graph neural networks for noisy graphs with sparse labels. In *Proceedings of the fifteenth ACM international conference on web search and data mining*, 2022.

[49] Mingze Dong and Yuval Kluger. Towards understanding and reducing graph structural noise for gnns. In *International Conference on Machine Learning*, 2023.