Graph Structural Attack by Perturbing Spectral Distance

Lu Lin
ll5fy@virginia.edu
University of Virginia
Charlottesville, VA 22904, USA

Ethan Blaser
ehb2bf@virginia.edu
University of Virginia
Charlottesville, VA 22904, USA

Hongning Wang
hw5x@virginia.edu
University of Virginia
Charlottesville, VA 22904, USA

ABSTRACT

Graph Convolutional Networks (GCNs) have fueled a surge of research interest due to their encouraging performance on graph learning tasks, but they are also shown vulnerability to adversarial attacks. In this paper, an effective graph structural attack is investigated to disrupt graph spectral filters in the Fourier domain, which are the theoretical foundation of GCNs. We define the notion of spectral distance based on the eigenvalues of graph Laplacian to measure the disruption of spectral filters. We realize the attack by maximizing the spectral distance and propose an efficient approximation to reduce the time complexity brought by eigen-decomposition. The experiments demonstrate the remarkable effectiveness of the proposed attack in both black-box and white-box settings for both test-time evasion attacks and training-time poisoning attacks. Our qualitative analysis suggests the connection between the imposed spectral changes in the Fourier domain and the attack behavior in the spatial domain, which provides empirical evidence that maximizing spectral distance is an effective way to change the graph structural property and thus disturb the frequency components for graph filters to affect the learning of GCNs.

CCS CONCEPTS

• **Computing methodologies** → *Learning latent representations*; **Neural networks**; • **Information systems** → *Data mining*.

KEYWORDS

Graph neural networks, adversarial attacks, graph spectral theory

ACM Reference Format:
Lu Lin, Ethan Blaser, and Hongning Wang. 2022. Graph Structural Attack by Perturbing Spectral Distance. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3534678.3539435

1 INTRODUCTION

Graph signal processing applies the idea of signal processing to graph data, allowing existing signal processing tools such as spectral filtering and sampling to be used for learning graph embeddings [10]. In particular, spectral filters are generalized to create Graph Convolutional Networks (GCNs), which have prominently advanced the state of the art on many graph learning tasks [18, 41]. However, despite their great success, recent works show that GCNs exhibit vulnerability to *adversarial perturbations*: such models can be easily fooled by small perturbations on graph structure or node properties, and thus generate inaccurate embeddings leading to erroneous predictions in downstream tasks [8, 47–50].

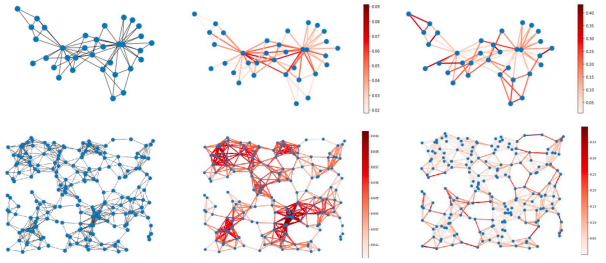
Graph data differs from image or text data due to the topological structure formed among nodes. On one hand, GCNs exploit such structures to aggregate information conveyed in nodes' neighborhoods, which yield better predictive power on many tasks (such as link prediction [36] and node classification [33]). But on the other hand, the complex dependency relations introduced by the topological structure of graphs also expose learning models to a greater risk: an attacker can mislead classifiers to erroneous predictions by just slightly perturbing the graph structure, without even modifying any node features. Various adversarial attacks have been studied on graph structure [15], considering different prediction tasks (node classification [43, 48, 50] or graph classification [8, 23]), attacker's knowledge (white-box [43, 48, 49] or black-box [8, 23]), attack phases (test-time evasion attack [5, 8, 48] or training-time poisoning attack [43, 50]), and perturbation types (edge modification [48, 50] or node modification [40]). In this paper, we focus on the structural attack by adding or removing edges to compromise the node classification performance of a victim GCN model.

Graph convolution, as the fundamental building block of GCNs, is designed to filter graph signals in the *Fourier* domain. Studies in spectral graph theory [7] show that the spectra (eigenvalues) of the graph Laplacian matrix capture graph structural properties (e.g., the second smallest eigenvalue, also known as the Fiedler value, reflects the algebraic connectivity of graphs [27]). Therefore exploiting spectral changes provides a comprehensive way to study the vulnerability of GCN models. However, so far most structural attack solutions only search for perturbations in the *spatial* domain. Ignoring the direct source of GCN models' vulnerability which resides in the Fourier domain limits the effectiveness of attacks.

Studying GCN models' vulnerability in the Fourier domain can effectively capture important edges that influence the structural property the most, e.g., the clustering structure of nodes. According to the concept of graph signal processing, the eigen-decomposition of the Laplacian matrix of a graph defines the frequency domain of message passing on the graph. Recent works have established the relationship between frequency components and graph clustering [11, 38]. Based on the ascending ordered eigenvalues of the Laplacian matrix, we can obtain both low- and high-frequency components, which play different roles in message passing on graphs. The eigenvectors associated with small eigenvalues carry smoothly varying signals, encouraging neighbor nodes to share similar properties (e.g., nodes within a cluster). In contrast, the eigenvectors associated with large eigenvalues carry sharply varying signals across edges (e.g., nodes from different clusters) [4, 14]. Figure 1 illustrates the concept on both the popularly studied social network

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08.
https://doi.org/10.1145/3534678.3539435



(a) Original graph (b) Low-frequency graph (c) High-frequency graph

Figure 1: Relationship between graph structural property and frequency components in Fourier domain on the Karate club graph (TOP) and a random geometric graph (BOTTOM). Color denotes the edge reconstructed when only using low-frequency components (b) or high-frequency components (c). Darker color indicates a larger edge reconstruction value.

graph Karate club and a random geometric graph. We visualize the edges by their reconstruction using the eigenvectors only associated with the top low- or high-frequency components in each graph respectively. For example, in Figure 1 (b), the color of each edge reflects its reconstruction only using eigenvectors associated with the lowest eigenvalues. On one hand, the information inside the closely connected components is retained when low-frequency components are used to reconstruct the graph; and on the other hand, the inter-cluster information is captured when constructing the graph with only high-frequency components. This example clearly illustrates that graph frequency components encode the global structure of graphs, which motivates us to study GCN models' vulnerability in the Fourier domain.

In this work, we propose a principled graph perturbation strategy in the Fourier domain to improve the effectiveness of adversarial attacks against GCN models. Specifically, we define the *spectral distance* between the original and perturbed graph, measured by the change in their Laplacian eigenvalues. Then we build a structural attack model which directly maximizes the spectral distance in a black-box fashion. To solve this combinatorial optimization problem, we relax the binary constraint on edge perturbation to a continuous one, and apply a randomization sampling strategy to generate valid binary edge perturbations. We name this method **SP**ectral **A**tta**C**k, abbreviated as **SPAC**. It is worth noting that generating the SPAC attack requires eigen-decomposition of the Laplacian matrix, which results in a time complexity of $O(n^3)$ with n nodes in a graph. To handle large graphs, we propose an approximation solution only based on a set of largest and smallest eigenvalues and their corresponding eigenvectors, and use eigenvalue perturbation theory [39] to avoid frequent computation of eigen-decomposition, which reduces the time complexity to $O(n^2)$. Our attack method is evaluated under both white-box and black-box settings for both evasion and poisoning attacks on a set of benchmark graph datasets. Promising empirical results demonstrate that convolutional graph learning models are sensitive to spectral changes, which expands the scope of adversarial attacks on graphs to the Fourier domain and opens up new possibilities to verify and enhance GCNs' robustness in both the spatial and Fourier domains.

2 RELATED WORK

Adversarial attacks on graph structures have been extensively studied in recent years. The vast majority of attack efforts manipulate graphs in the spatial domain to maximize a task-specific attack objective. However, the vulnerability of graph convolutions in the Fourier domain is less studied in existing attack solutions. We bridge the gap by measuring and maximizing the spectral changes in the graph Laplacian matrix, such that we can directly disrupt the graph spectral filters and attack the graph convolutions.

Adversarial attack on graph structures. The attacker aims to perturb the graph adjacency matrix in directions that lead to large classification loss. In the white-box setting, the attacker follows the gradients on the adjacency matrix to find such perturbations [6, 20, 44, 47–50]. Different strategies are exploited to convert continuous gradients into binary edge modifications. Topology attack [24] uses randomization sampling to select sub-optimal binary perturbations. Nettack [49] and FGA [6] select edge changes with the largest gradient greedily. Metattack [50] first calculates meta-gradient on graph adjacency matrix to solve a bi-level optimization problem for poisoning attack, and then greedily picks perturbations with the largest meta-gradient. In the black-box setting, the attacker cannot access gradients of the victim model but uses a proxy (e.g. model output scores) to search for the best perturbations [8, 22, 23]. Reinforcement learning based solutions [5, 8, 23, 29] make a series of edge addition or deletion decisions that yield the maximal attack utility and thus can serve for black-box setting.

These attacks search for perturbations in the spatial space, but the target GCNs generate node embeddings by the signal filters defined in the Fourier space. Thus the vulnerability of graph convolutions reflected on the graph spectral changes cannot be fully realized. Our method captures such vulnerability directly in the Fourier domain measured by the spectral distance between the original and perturbed graphs for a more effective attack.

Spectral perturbations on graphs. Existing attack methods in the Fourier space are generally sparse. Bojchevski and Günnemann [2] reformulate random walk based models as a matrix factorization problem, and propose an attack strategy to search for edges that lead to large eigenvalue changes in the derived matrix. However, this method is model-specific and cannot be easily applied to general forms of GCNs. GF-Attack [5] constructs an objective based on the low-rank graph spectra and feature matrix to guide the attack in a black-box fashion. A universal attack on deformable 3D shape data is proposed to change the scale of its eigenvalues [30], but it is not studied in the graph domain. DICE [45] corrupts the graph structure by "deleting edges internally and connecting nodes externally" across clusters which implicitly influences the graph's spectral property. But this heuristic is performed without any principled guidance. Studies that analyze spectral graph filters [16, 17, 19] provide the theoretical stability upper bounds of popular filters used in graph convolution models, such as polynomial and identity filters. It is shown that the filters become unstable if the end nodes of changed edges have low degrees or the perturbation is concentrated spatially around any single node [17]. Our method empirically shows that we can attack the vulnerability of these filters and break such requirements by directly maximizing graph spectral changes.

3 SPECTRAL ATTACK ON GRAPHS

In this section, we first briefly discuss the spectral graph filters which are the key building blocks of graph convolution models. We then propose to maximize the changes on the graph Laplacian spectrum, such that we can exploit the edge perturbation budget to most effectively influence the spectral filters and attack graph convolutions. We solve the resulting optimization problem using gradient descent, and propose an efficient approximation via eigenvalue perturbation theory on selective eigenvalues. We finally discuss the theoretical evidence showing the dependency between the eigenvalues of graph Laplacian and the stability of GCN models, which supports the proposed spectral attack on graph data.

3.1 Preliminaries

Notations. Let $G = (V, E)$ be a connected undirected graph with n nodes and m edges. Let $\mathbf{A} \in \{0,1\}^{n\times n}$ be its adjacency matrix. The diagonal degree matrix can be calculated by $\mathbf{D} = \text{diag}(\mathbf{A}\mathbf{1}_n)$ with entry $\mathbf{D}_{ii} = d_i = \sum_{j=1}^{n} \mathbf{A}_{ij}$, and $\mathbf{1}_n$ is an all-one vector with dimension n. The normalized Laplacian matrix of a graph is defined as $\mathbf{L} = \text{Laplacian}(\mathbf{A}) = \mathbf{I}_n - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, where $\mathbf{I}_n$ is an $n \times n$ identity matrix. Since $\mathbf{L}$ is symmetric positive semi-definite, it admits an eigen-decomposition $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$. The diagonal matrix $\mathbf{\Lambda} = \text{eig}(\mathbf{L}) = \text{diag}(\lambda_1, \ldots, \lambda_n)$ consists of the real eigenvalues of $\mathbf{L}$ in an increasing order such that $0 = \lambda_1 \leq \cdots \leq \lambda_n \leq 2$, and the corresponding $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_n] \in \mathbb{R}^{n\times n}$ is a unitary matrix where the columns consist of the eigenvectors of $\mathbf{L}$. $\mathbf{X} \in \mathbf{R}^{n\times d}$ denotes the node feature matrix where each node v is associated with a d-dimensional feature vector.

Graph Fourier transform. By viewing graph embedding models from a signal processing perspective, the normalized Laplacian $\mathbf{L}$ serves as a shift operator and defines the frequency domain of a graph [35]. As a result, the eigenvectors of $\mathbf{L}$ can be considered as the graph Fourier bases, and the eigenvalues correspond to frequency components. Take one column of $\mathbf{X}$ as an example of graph signal, which can be compactly represented as $\mathbf{x} \in \mathbb{R}^n$. The graph Fourier transform of graph signal $\mathbf{x}$ is given by $\hat{\mathbf{x}} = \mathbf{U}^\top\mathbf{x}$ and the inverse graph Fourier transform is then $\mathbf{x} = \mathbf{U}\hat{\mathbf{x}}$. The graph signals in the Fourier domain are filtered by amplifying or attenuating the frequency components $\mathbf{\Lambda}$.

Spectral graph convolution. At the essence of different graph convolutional models is the spectral convolution, which is defined as the multiplication of signal $\mathbf{x}$ with a filter g_ϕ parameterized by $\phi \in \mathbb{R}^n$ in the Fourier domain [9]:

$$g_\phi(\mathbf{L}) \star \mathbf{x} = \mathbf{U}g^*_\phi(\mathbf{\Lambda})\mathbf{U}^\top\mathbf{x} \tag{1}$$

where the parameter ϕ is a vector of spectral filter coefficients. The filter g_ϕ defines a smooth transformation function, and a commonly used filter is the polynomial filter:

$$g^*_\phi(\mathbf{\Lambda}) = \sum_{k=0}^{\infty} \phi_k \mathbf{\Lambda}^k \tag{2}$$

which can be approximated by a truncated expansion. A commonly adopted approximation is based on the Chebyshev polynomials $T_k(\mathbf{\Lambda})$, which are recursively defined as $T_0(\mathbf{\Lambda}) = \mathbf{I}_n, T_1(\mathbf{\Lambda}) = \mathbf{\Lambda}$ and $T_{k+1}(\mathbf{\Lambda}) = 2\mathbf{\Lambda}T_k(\mathbf{\Lambda}) - T_{k-1}(\mathbf{\Lambda})$. Using the Chebyshev polynomials up to the K-th order achieves the following approximation [13]:

$$g^*_\phi(\mathbf{\Lambda}) \approx \sum_{k=0}^{K} \phi_k T_k(\tilde{\mathbf{\Lambda}}) \tag{3}$$

with a rescaled $\tilde{\mathbf{\Lambda}} = 2\mathbf{\Lambda}/\lambda_n - \mathbf{I}_n$.

Graph Convolutional Network (GCN). A vanilla GCN is a first-order approximation of the spectral graph convolution with the Chebyshev polynomials [18]. Setting $K = 1$, $\phi_0 = -\phi_1$ in Eq. (3) and approximating $\lambda_n \approx 2$, we obtain the convolution operation $g_\phi(\mathbf{L}) \star \mathbf{x} = (\mathbf{I}_n + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{x}$. We can replace matrix $\mathbf{I}_n + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ with a self-loop enhanced version $\tilde{\mathbf{L}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$ and $\tilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}_n$. This resembles the vanilla GCN layer with activation function σ and trainable network parameters $\mathbf{\Theta}$ for feature transformation:

$$\mathbf{H}^{(l+1)} = \sigma\left(\tilde{\mathbf{L}}\mathbf{H}^{(l)}\mathbf{\Theta}^{(l)}\right) \tag{4}$$

where the signals from the previous layer $\mathbf{H}^{(l)}$ is filtered to generate new signals $\mathbf{H}^{(l+1)}$. To unify the notations, $\mathbf{H}^{(0)} = \mathbf{X}$ denotes the input node features, and $\mathbf{Z} = \mathbf{H}^{(L)}$ denotes the output node embeddings of an L-layer GCN model.

3.2 Spectral Distance on Graphs

Based on the aforementioned spectral perspective for understanding GCNs, we aim to generate edge perturbations that can disrupt the spectral filters the most when processing input signals on the graph. We measure the disruption by the changes in the eigenvalues of graph Laplacian, which we define as the spectral distance.

As shown in Eq. (1), the spectral filters $g^*_\phi(\mathbf{\Lambda})$ are the key in graph convolutions to encode graph signals that are transformed in the Fourier domain. The output of the spectral filters is then transformed back to the spatial domain to generate node embeddings. Therefore, perturbing the spectral filters $g^*_\phi(\mathbf{\Lambda})$ will affect the filtered graph signals and produce inaccurate node embeddings. To measure the changes in spectral filters, we define the spectral distance between the original graph G and perturbed graph G' as:

$$\mathcal{D}_{\text{spectral}} = \|g^*_\phi(\mathbf{\Lambda}) - g^*_\phi(\mathbf{\Lambda}')\|_2 \tag{5}$$

where $\mathbf{\Lambda}$ and $\mathbf{\Lambda}'$ are the eigenvalues of the normalized graph Laplacian for G and G' respectively. The spectral distance $\mathcal{D}_{\text{spectral}}$ is determined by both filter parameters ϕ and the frequency components $\mathbf{\Lambda}$. For graph embedding models based on the vanilla GCN [18], we follow their design of spectral filters which uses the first-order approximation of the Chebyshev polynomials in Eq. (3) and sets $\phi_0 = -\phi_1 = 1$, which gives:

$$g^*_\phi(\mathbf{\Lambda}) \approx \phi_0\mathbf{I}_0 + \phi_1\mathbf{\Lambda} = \mathbf{I}_n - \mathbf{\Lambda} \tag{6}$$

Plugging it into Eq. (5), we conclude the following spectral distance which is only related to the eigenvalues of graph Laplacian:

$$\begin{aligned}\mathcal{D}_{\text{spectral}} &\approx \|(\mathbf{I}_n - \mathbf{\Lambda}) - (\mathbf{I}_n - \mathbf{\Lambda}')\|_2 = \|\mathbf{\Lambda} - \mathbf{\Lambda}'\|_2 \\ &= \|\text{eig}(\text{Laplacian}(\mathbf{A})) - \text{eig}(\text{Laplacian}(\mathbf{A}'))\|_2 \end{aligned} \tag{7}$$

This spectral distance reflects the changes of spectral filters due to the graph perturbation. Therefore, if we perturb the graph by directly maximizing the spectral distance, we can impose the most effective changes to graph filters and thus disrupt the generated node embeddings the most.

3.3 Spectral Attack on Graph Structure

Since the spectral distance measures the changes of the spectral filters on the graph after perturbation, we can produce effective attacks by maximizing the resulting spectral distance. In this section, we first show the general formulation of spectral attack on graph structure; then we propose a practical approach to solve the problem efficiently; finally we extend the proposed attack to existing white-box attack frameworks.

Structural perturbation matrix. The goal is to find the perturbation on the graph adjacency matrix that can maximize the spectral distance $\mathcal{D}_{\text{spectral}}$ defined in Eq. (7). We first define the structural perturbation as a binary perturbation matrix $\mathbf{B} \in \{0,1\}^{n\times n}$, which indicates where to flip the edges in G. The new adjacency matrix after the perturbation is then a function of the perturbation matrix, which can be obtained as follows [48]:

$$g(\mathbf{A}, \mathbf{B}) = \mathbf{A} + \mathbf{C} \circ \mathbf{B},\ \mathbf{C} = \bar{\mathbf{A}} - \mathbf{A} \tag{8}$$

where $\bar{\mathbf{A}}$ is the complement matrix of the adjacency matrix $\mathbf{A}$, calculated by $\bar{\mathbf{A}} = \mathbf{1}_n\mathbf{1}_n^\top - \mathbf{I}_n - \mathbf{A}$, with $(\mathbf{1}_n\mathbf{1}_n^\top - \mathbf{I}_n)$ denoting the fully-connected graph without self loops. Therefore, $\mathbf{C} = \bar{\mathbf{A}} - \mathbf{A}$ denotes legitimate addition or deletion operations on each node pair: adding an edge is allowed between node i and j if $\mathbf{C}_{ij} = 1$, and removing an edge is allowed if $\mathbf{C}_{ij} = -1$. Taking the Hadamard product $\mathbf{C} \circ \mathbf{B}$ finally gives valid edge perturbations to the graph.

Spectral attack. To generate an effective structural attack, we seek a perturbation matrix $\mathbf{B}$ that maximizes the spectral distance $\mathcal{D}_{\text{spectral}}$ defined in Eq. (7). More specifically, given a finite budget of edge perturbation, e.g., $\|\mathbf{B}\|_0 \le \epsilon|E|$ with $|E|$ denoting the number of edges, we formulate the **SP**ectral **A**tta**C**k (**SPAC**) as the following optimization problem:

$$\max_{\mathbf{B}} \mathcal{L}_{\text{SPAC}} \coloneqq \mathcal{D}_{\text{spectral}} \tag{9}$$
$$\text{subject to } \|\mathbf{B}\|_0 \le \epsilon|E|, \mathbf{B} \in \{0,1\}^{n\times n}, \mathbf{A}' = g(\mathbf{A}, \mathbf{B})$$

which is not straightforward to solve because of two challenges: 1) it is a combinatorial optimization problem due to the binary constraint on $\mathbf{B}$; 2) the objective involves eigen-decomposition of the Laplacian matrix, which is time-consuming especially for large graphs. Next, we introduce our practical solution to address these challenges such that we can efficiently generate the structural perturbations for the spectral attack.

3.4 Implementation of SPAC

In this section, we discuss our solution for the combinatorial optimization problem involving eigen-decomposition in Eq. (9). Specifically, we first relax the combinatorial problem and use a randomization sampling strategy to generate the binary perturbation matrix; we then introduce an approximation strategy to reduce the complexity of backpropagation through eigen-decompostion.

Binary perturbation via gradient descent. For the ease of optimization, we relax $\mathbf{B} \in \{0,1\}^{n\times n}$ to its convex hull $\mathbf{\Delta} \in [0,1]^{n\times n}$ [48], which simplifies the combinatorial problem in Eq. (9) to be the following continuous optimization problem:

$$\max_{\mathbf{\Delta}} \mathcal{L}_{\text{SPAC}} \coloneqq \mathcal{D}_{\text{spectral}} \tag{10}$$
$$\text{subject to } \|\mathbf{\Delta}\|_1 \le \epsilon|E|, \mathbf{\Delta} \in [0,1]^{n\times n}, \mathbf{A}' = g(\mathbf{A}, \mathbf{\Delta})$$

which can be solved via gradient descent. Applying chain rule, we calculate the gradient with respect to $\mathbf{\Delta}$ as follows:

$$\frac{\partial \mathcal{L}_{\text{SPAC}}}{\partial \mathbf{\Delta}_{ij}} = \sum_{k=1}^{n} \frac{\partial \mathcal{L}_{\text{SPAC}}}{\partial \lambda'_k} \sum_{p=1}^{n}\sum_{q=1}^{n} \frac{\partial \lambda'_k}{\partial \mathbf{L}'_{pq}} \frac{\partial \mathbf{L}'_{pq}}{\partial \mathbf{\Delta}_{ij}} \tag{11}$$

Recall that $\mathbf{L}' = \text{Laplacian}(\mathbf{A}')$ and λ'_k is an eigenvalue of $\mathbf{L}'$ in Eq. (7). We now focus on the gradient calculation that involves eigen-decomposition. Since the gradient calculation on the rest is straightforward, we leave it to the appendix. For a real and symmetric matrix $\mathbf{L}$, one can obtain the derivatives of its eigenvalue λ_k and eigenvector $\mathbf{u}_k$ by: $\partial\lambda_k/\partial\mathbf{L} = \mathbf{u}_k\mathbf{u}_k^\top, \partial\mathbf{u}_k/\partial\mathbf{L} = (\lambda_k\mathbf{I} - \mathbf{L})^+\mathbf{u}_k$ [25, 32]. Therefore, we can directly obtain the closed-form derivative of the eigenvalues in Eq (11) as: $\partial\lambda'_k/\partial\mathbf{L}'_{pq} = \mathbf{u}'_{kp}\mathbf{u}'_{kq}$. Note that the derivative calculation requires distinct eigenvalues. This does not hold for graphs satisfying *automorphism*, which reflects structural symmetry of graphs [12]. To avoid such cases, we add a small noise term to the adjacency matrix of the perturbed graph[1], e.g., $\mathbf{A}' + \varepsilon \times (\mathbf{N} + \mathbf{N}^\top)/2$, where each entry in $\mathbf{N}$ is sampled from a uniform distribution $\mathcal{U}(0,1)$ and ε is a very small constant. Such a noise addition will almost surely break the graph automorphism, thus enable a valid gradient calculation of eigenvalues.

Solving the relaxed problem in Eq. (10) using projected gradient descent gives us a continuous perturbation matrix $\mathbf{\Delta}$ that maximizes the spectral change. To recover valid edge perturbations from the continuous $\mathbf{\Delta}$, we then generate a near-optimal solution for the binary perturbation matrix $\mathbf{B}$ via the randomization sampling strategy [48]. Specifically, we use $\mathbf{\Delta}$ as a probabilistic matrix to sample the binary assignments as follows:

$$\mathbf{B}_{ij} = \begin{cases} 1, & \text{with probability } \mathbf{\Delta}_{ij} \\ 0, & \text{with probability } 1 - \mathbf{\Delta}_{ij} \end{cases} \tag{12}$$

Time complexity analysis. Suppose we take aforementioned projected gradient descent for T steps. For each step, SPAC takes eigen-decomposition with time complexity $O(n^3)$ and samples binary solution with $O(n^2)$ edge flips. The overall time complexity for solving SPAC is $O(Tn^3 + Tn^2)$, which is mainly attributed to eigen-decomposition and is considerably expensive for large graphs. Next, we discuss an approximation solution to improve its efficiency.

Efficient approximation for SPAC. To reduce the computation cost of eigen-decomposition, instead of measuring the spectral distance over all the frequency components, we decide to only maintain the k_1 lowest- and k_2 highest-frequency components which are the most informative, as suggested by the spectral graph theory (this can also be intuitively observed in Figure 1). Specifically, $\mathcal{D}_{\text{spectral}}$ in Eq. (7) can be approximated as follows:

$$\mathcal{D}_{\text{spectral-approx}} = \sqrt{\sum_{i\in\mathcal{S}} (\lambda_i - \lambda'_i)^2} \tag{13}$$

where $\mathcal{S} = \{1, \ldots, k_1, n-k_2, \ldots, n\}$. This reduces the time complexity $O(n^3)$ for exact eigen-decomposition to $O((k_1 + k_2)\cdot n^2)$ for the corresponding selective eigen-decomposition using the Lanczos Algorithm [28]. To avoid frequent computation of the selective eigenvalues and further improve efficiency, we propose to estimate

[1]The form of $(\mathbf{N} + \mathbf{N}^\top)/2$ is to keep the perturbed adjacency matrix symmetric for undirected graphs.

the change in each eigenvalue for any edge perturbation based on the eigenvalue perturbation theory [2, 5, 31].

Theorem 1. *Let $\mathbf{u}_i$ be the i-th generalized eigenvector of $\mathbf{L}$ with generalized eigenvalue λ_i. Let $\mathbf{L}' = \mathbf{L} + \nabla\mathbf{L}$ be the perturbed Laplacian matrix, and $\mathbf{M}'$ be the diagonal matrix summing over rows of $\mathbf{L}'$. The generalized eigenvalue λ_i' of the Laplacian matrix $\mathbf{L}'$ that solves $\mathbf{L}'\mathbf{u}_i' = \lambda_i'\mathbf{M}'\mathbf{u}_i'$ is approximately $\lambda_i' \approx \lambda_i + \nabla\lambda_i$ with:*

$$\lambda_i - \lambda_i' = \nabla\lambda_i = \mathbf{u}_i^\top(\nabla\mathbf{L} - \lambda_i diag(\nabla\mathbf{L}\cdot\mathbf{1}_n))\mathbf{u}_i \tag{14}$$

The proof is given in the appendix. Instead of recalculating the eigenvalues λ_i' for the updated $\mathbf{L}'$ in each step when measuring the spectral distance, we use this theorem to approximate the change of each eigenvalue in Eq. (13) in linear time. Suppose we execute Eq. (14) to calculate spectral distance in Eq. (13) for m steps and compute the exact eigenvalues every m step to avoid error accumulation, we can achieve an overall time complexity $O\big((1+\frac{k_1+k_2}{m})Tn^2\big)$. We name the spectral attack equipped with the approximation stated in Eq. (13) and (14) as **SPAC-approx**.

Algorithm 1 summarizes the implementation of SPAC (in line 5) and its approximation SPAC-approx (in line 7-10). After obtaining the objective function based on the (approximated) spectral distance, the algorithm further updates the continuous perturbation matrix $\mathbf{\Delta}$ via gradient descent and finally generates the binary structural perturbation matrix $\mathbf{B}$ by sampling from $\mathbf{\Delta}$.

Algorithm 1 Spectral Attack on Graph Structure

Input: $G = (\mathbf{X}, \mathbf{A})$; total step T; step size η; k_1, k_2, m.

1: Initialize continuous perturbation $\mathbf{\Delta}_0 \in [0,1]^{n\times n}$
2: Initialize perturbed Laplacian matrix $\mathbf{L}' = \text{Laplacian}(g(\mathbf{A}, \mathbf{\Delta}))$
3: **for** $t = 0, \ldots, T-1$ **do**
4: **if** SPAC **then**
5: $\mathcal{L}(\mathbf{\Delta}) = \|\mathbf{\Lambda} - \mathbf{\Lambda}'\|_2$ by Eq. (10), with $\mathbf{\Lambda}' = \text{eig}(\mathbf{L}')$
6: **else if** SPAC-approx **then**
7: **if** $t \,\%\, m = 0$ **then**
8: $\mathcal{L}(\mathbf{\Delta}) = \sqrt{\sum_{i\in\mathcal{S}}(\lambda_i - \lambda_i')^2}$ by Eq. (13)
9: **else**
10: $\mathcal{L}(\mathbf{\Delta}) = \sqrt{\sum_{i\in\mathcal{S}}(\mathbf{u}_i^\top(\nabla\mathbf{L} - \lambda_i \text{diag}(\nabla\mathbf{L}\cdot\mathbf{1}_n))\mathbf{u}_i)^2}$ by Eq. (14) and Eq. (13), with $\nabla\mathbf{L} = \mathbf{L}' - \mathbf{L}$
11: **end if**
12: **end if**
13: Compute gradient on $\mathbf{\Delta}$: $\mathbf{g}_t = \nabla\mathcal{L}(\mathbf{\Delta})$ by Eq. (11)
14: Update $\mathbf{\Delta}_{t+1} = \mathbf{\Delta}_t + \eta\cdot\mathbf{g}_t$
15: Project $\mathbf{\Delta}_{t+1}$ to its convex hull $\mathbf{\Delta}_{t+1} \in [0,1]^{n\times n}$
16: **end for**
17: Output binary perturbation $\mathbf{B}$ by sampling from $\mathbf{\Delta}_T$ via Eq. (12)

3.5 Extension in White-box Setting

The proposed attack only requires information about graph spectrum, therefore it can be conducted alone in the black-box setting as Eq. (9) stated. Since SPAC does not rely on any specific embedding model, it can also serve as a general recipe for the white-box attack setting. Next, we show how to easily combine SPAC with white-box attack models.

Victim Graph Embedding Model. Without loss of generality, we consider the vanilla GCN model for the node classification task. Given a set of labeled nodes $V_0 \subset V$, where each node i belongs to a class in a label set $y_i \in Y$. The GCN model aims to learn a function f_θ that maps each node to a class. We consider the commonly studied transductive learning setting, where the test (unlabeled) nodes with associated features and edges are observed in the training phase. The GCN model is trained by minimizing the following loss function:

$$\min_\theta \ \mathcal{L}_{\text{train}}(f_\theta(G)) = \sum_{v_i\in V_0} \ell(f_\theta(\mathbf{A},\mathbf{X})_i, y_i)$$

where $f_\theta(\mathbf{X},\mathbf{A})_i$ and y_i are the predicted and ground-truth labels of node v_i and $\ell(\cdot,\cdot)$ is a loss function of choice, such as the cross entropy loss.

White-box Spectral Attack. We have shown that the changes of the spectral filters are essential for attackers to disrupt graph convolutions, thus we propose to maximize the spectral distance between the original graph and the perturbed one. In the meanwhile, maximizing the task-specific attack objective is necessary to achieve the attack's goal to compromise the prediction performance. To generate edge perturbations that lead to both disrupted spectral filters and erroneous classifications, we propose to maximize the spectral distance and task-specific attack objective simultaneously. Specifically, given the test node-set $V_t \subset V$, the attack model aims to find the edge perturbation $\mathbf{B}$ that solves the following optimization problem:

$$\begin{aligned}\max_{\mathbf{B}} \quad & \underbrace{\sum_{v_i\in\mathcal{V}_t} \ell_{\text{atk}}(f_{\theta^*}(\mathbf{A}',\mathbf{X})_i, y_i)}_{\text{task-specific attack objective } \mathcal{L}_{\text{attack}}} + \beta\cdot\mathcal{L}_{\text{SPAC}} \\ \text{subject to} \quad & \|\mathbf{B}\|_0 \le \epsilon, \mathbf{B}\in\{0,1\}^{n\times n}, \mathbf{A}' = g(\mathbf{A},\mathbf{B}) \\ & \theta^* = \arg\min_\theta \ \mathcal{L}_{\text{train}}(f_\theta(\hat{G}))\end{aligned} \tag{15}$$

where the third constraint controls when to apply the attack: setting $\hat{G} = G$ makes it an evasion attack, so that the graph embedding model training is not affected; and setting $\hat{G} = G'$ makes it poisoning attack with perturbed training data. The attack objective $\mathcal{L}_{\text{attack}}$ is a flexible placeholder that can adapt to many loss designs, for a simple example, the cross-entropy loss on test nodes in the node classification task. The hyper-parameter β balances the effect of these two components, which is set based on graph properties such as edge density. Algorithm 1 also applies for the white-box setting by plugging in $\mathcal{L}_{\text{attack}}$ to the objective function. We will discuss the choices of attack objectives and hyper-parameters in the experiment section for our empirical evaluations.

3.6 Discussion

Our spectral attack is based on the fact that the spectral filters are the fundamental building blocks for graph convolutions to process graph signals in the Fourier domain. Therefore, searching the graph perturbations in the direction that causes the most changes in the spectral filters, measured by eigenvalues of graph Laplacian, is expected to best disrupt graph convolutions. This is also supported by recent theoretical evidence in the field.

Some recent literature has shown that the stability of GCN models is closely related to the eigenvalues of the graph Laplacian. For example, it is proved that the generalization gap of a single layer

Table 1: Dataset statistics. D is the node feature dimension ("-" means no node feature). K is the number of classes.

Dataset	#Node	#Edge	Density	D	K
Cora	2,708	5,278	0.0014	1433	7
Citeseer	3,312	4,536	0.0008	3703	6
Polblogs	1,490	16,715	0.015	-	2
Blogcatalog	5,196	171,743	0.013	8189	6

Table 2: Average running time (in seconds) for 10 runs of evasion attack with $T = 100$ and $\epsilon = 0.05$.

Datasets	Random	DICE	GF-Attack	**SPAC**	**SPAC-approx**
Cora	0.05	55.58	66.73	212.53	75.46
Citeseer	0.06	46.72	57.22	116.07	60.93
Polblogs	0.02	14.84	21.73	44.18	22.98
Blogcatalog	1.46	127.72	132.23	352.52	147.34

GCN model f_θ trained via T-step SGD is $O(\lambda_n^{2T})$, where λ_n is the largest eigenvalue of graph Laplacian [42]. Meanwhile, Weinberger et al. [46] proved that a generalization estimate is *inversely* proportional to the second smallest eigenvalue of the graph Laplacian λ_2. These findings suggest that manipulating the graph by perturbing the eigenvalues can potentially aggravate the generalization gap of GCN, causing a larger generalization error.

4 EXPERIMENTS

We performed extensive evaluations of the proposed spectral attack on four popularly used graph datasets, where we observed remarkable improvements in the attack's effectiveness. This section summarizes our experiment setup, performance on both evasion and poisoning attacks, and qualitative analysis on the perturbed graphs to study the effect of the spectral attack.

4.1 Setup

Datasets. We evaluated the proposed attack on two citation network benchmark datasets, Cora [26] and Citeseer [37], as well as two social network datasets, Polblogs [1] and Blogcatalog [34]. Table 1 summarizes the statistics of these datasets. We followed the transductive semi-supervised node classification setup in [18], where only 20 sampled nodes per class were used for training, but the features and edges of all nodes were visible to the attacker during the training stage. The predictive accuracy of the trained classifier was evaluated on 1000 randomly selected test nodes. The evasion attacker can query the trained classifier, but cannot access the training nodes; the poisoning attacker can observe the training nodes, but cannot access the labels of the test nodes.

Baseline attacks. We compared the proposed attack model SPAC against three attacks in the black-box setting, and further verified the effectiveness of SPAC by combining it with five baselines in white-box setting[2]. Black-box baselines for both *evasion* and *poisoning* attack include: 1) **Random** directly attacks the graph structure by randomly flipping the edges; 2) **DICE** [45] is a heuristic method that deletes edges internally and connects nodes externally across class clusters; 3) **GF-Attack** [5] perturbs the structure by maximizing the loss of low-rank matrix approximation defined over small eigenvalues. We further evaluate the performance of SPAC combined with white-box attack baselines which include: 1) **PGD-CE** [48] is an *evasion* attack which maximizes the cross-entropy (CE) loss on the test nodes via projected gradient descent (PGD) algorithm [24]; 2) **PGD-C&W** [48] is an *evasion* attack which perturbs edges by minimizing the C&W score, which is the margin between the largest and the second-largest prediction score, defined by Carlini-Wagner attack [3]; 3) **Max-Min** [48] is a *poisoning* attack, which solves the bi-level optimization problem by iteratively generating structural perturbations (to maximize the cross-entropy loss) and retraining a surrogate victim model on the perturbed graph (to minimize the loss); 4) **Meta-Train** [50] is a *poisoning* attack which uses meta-gradients on the perturbation matrix to maximize the training classification loss; 5) **Meta-Self** [50] is a *poisoning* attack that extends Meta-Train to maximize the self-training loss on test nodes using the predicted labels;

Variants of SPAC. The proposed attack can be realized with exact and approximated spectral distance, which gives **SPAC** and **SPAC-approx**. We will compare their attack performance and running time. Meanwhile, adopting the objectives from white-box baselines to Eq. (15) generates the following white-box attack variants: 1) **SPAC-CE** is an *evasion* attack that jointly maximizes the cross-entropy loss and spectral distance; 2) **SPAC-C&W** is an *evasion* attack combining the *negative* C&W score and spectral distance; 3) **SPAC-Min** extends Max-Min by maximizing the loss as in SPAC-CE for poisoning attack; 4) **SPAC-Train** includes the spectral distance

[2]We conducted the comparative experiments using DeepRobust Library [21].

Table 3: Misclassification rate (%) with $\epsilon = 0.05$ for evasion attack (upper rows) and poisoning attack (lower rows).

Stage	Attack	Cora	Citeseer	Polblogs	Blogcatalog
Evasion	Clean	0.184	0.295	0.128	0.276
	Random	0.189	0.301	0.153	0.280
	DICE	0.205	0.308	0.202	0.329
	GF-Attack	0.198	0.311	0.179	0.333
	SPAC	**0.220**	**0.314**	**0.212**	**0.354**
	SPAC-approx	0.212	0.305	0.208	0.341
	PGD-CE	0.237	0.349	0.167	0.441
	SPAC-CE	**0.255**	**0.352**	**0.188**	**0.458**
	PGD-C&W	0.249	0.388	0.216	0.447
	SPAC-C&W	**0.260**	**0.395**	**0.229**	**0.464**
Poison	Clean	0.184	0.295	0.128	0.276
	Random	0.189	0.309	0.126	0.277
	DICE	0.207	0.310	**0.246**	0.306
	GF-Attack	0.195	0.306	0.202	0.334
	SPAC	**0.222**	**0.338**	0.234	**0.478**
	SPAC-approx	0.215	0.322	0.220	0.454
	Max-Min	0.240	0.359	0.167	0.489
	SPAC-Min	**0.255**	**0.375**	**0.188**	**0.504**
	Meta-Train	**0.290**	0.392	0.274	0.360
	SPAC-Train	0.285	**0.412**	**0.298**	**0.377**
	Meta-Self	0.427	0.499	**0.478**	0.590
	SPAC-Self	**0.489**	**0.508**	0.472	**0.599**

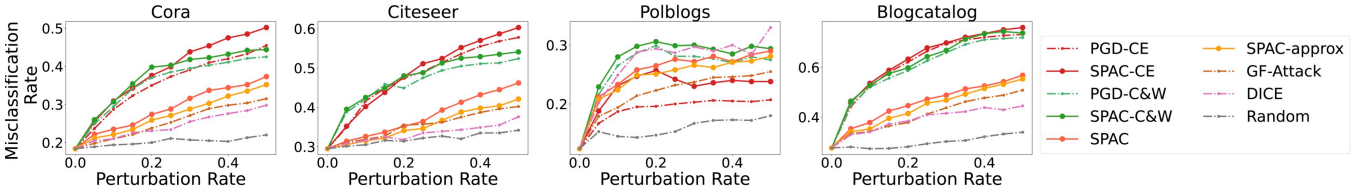


Figure 2: Misclassification rate under different perturbation rates for evasion attack.

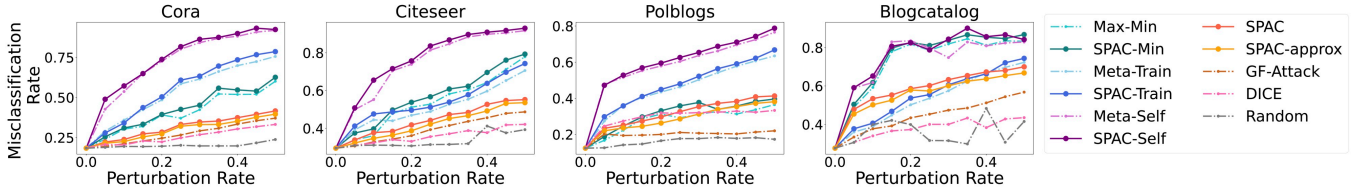


Figure 3: Misclassification rate under different perturbation rates for poisoning attack.

to Meta-Train for the meta-gradient calculation; 5) **SPAC-Self** enhances the loss of Meta-Train by the spectral distance. The detailed objective for each variant is summarized in the appendix.

Hyper-parameters. We adopt a two-layer GCN as the victim classifier, whose hidden dimension of the first layer is 64 and that of the second layer is the number of classes K on each dataset. The setup for total step T and step size η is summarized in the appendix. For SPAC-approx, we set $k_1 = 128$, $k_2 = 64$ and $m = 10$. In the white-box setting, the hyper-parameter β controls the weight of the spectral distance in the overall attack objective. Since the spectrum reflects the global property of the graph, the weight should be tuned based on the graph statistics. Empirically, we find that setting β proportional to the density of graph is effective. Specifically, according to the density of each dataset listed in Table 1, we set $\beta = 1.4$ for Cora network, $\beta = 0.8$ for Citeseer, $\beta = 13.0$ for Blogcatalog, and $\beta = 15.0$ for Polblogs. The sensitivity of hyper-parameters is discussed in Section 4.3. All experiments were conducted on RTX2080Ti GPUs.

4.2 Structural Attack Performance

Performance in evasion attack. In the evasion attack setting, we first trained a GCN classifier on the small training set V_0 with a clean graph $\hat{\mathcal{G}} = \mathcal{G}$. Then the classifier was fixed, and the attackers generated edge perturbations based on the classifier's predictions on the test nodes. Table 3 summarizes the misclassification rates under $\epsilon = 0.05$, which allows 5% edges to be perturbed. An extensive comparison with different perturbation rates is provided in Figure 2, where the solid lines with darker color denote SPAC variants while the dashed lines with lighter color represent baseline attacks.

In the black-box setting, randomly flipping edges (Random) cannot effectively influence the classifier's overall performance. DICE provides an effective attack by leveraging the label information. GF-Attack undermines the performance of GCNs by attacking its low-rank approximation. Our methods, both SPAC and SPAC-approx, disrupt the overall spectral filters and achieve the largest misclassification rate. This shows the effectiveness of the proposed attack principle based on the spectral distance, which reveals the essence of vulnerability in graph convolutions.

Second in the white-box setting, SPAC-CE and SPAC-C&W stand in stark contrast to PGD-CE and PGD-C&W: we can observe a remarkable improvement introduced by SPAC in the misclassification rate. The evasion attack results confirm that maximizing the spectral distance can considerably disrupt the trained classifier by changing the graph frequencies in the Fourier domain and invalidating the spectral filters.

Performance in poisoning attack. In the poisoning attack setting, we can only indirectly affect the classifier by perturbing the training graph structure. We generated the edge perturbations, and then used the poisoned structure to train the victim GCN model and reported its misclassification rate on test nodes in a clean graph. From Table 3 and Figure 3, we can again verify the effectiveness of the proposed spectral attack. Under the black-box setting, SPAC and SPAC-approx are the most effective attacks in most cases. Under the white-box setting, Max-Min only accesses training nodes to perturb the graph without querying test nodes. Meta-Train calculates the meta-gradient on training nodes to capture the change of loss after retraining the surrogate GCN model. Meta-Self instead does not use the training nodes, but only queries CGN's prediction scores on test nodes. Among baselines, the Meta-Self attack is shown to be the most effective, which is expected, because the current semi-supervised setting provides a much larger set of unlabeled nodes that can be fully used by Meta-Self. Overall, our attack based on the spectral distance still brought in a clear improvement to the misclassification rate across different datasets and attack methods.

Computational efficiency. We empirically evaluated the efficiency of SPAC and SPAC-approx in Table 2, which compares the average running time of 10 runs for evasion attack. Our proposed SPAC-approx can achieve a comparable efficiency as GF-Attack. Combining with the attack performance, SPAC-approx is verified to be an effective and efficient structural attack.

4.3 Analysis of SPAC

Given the empirical effectiveness of the proposed attack strategy, we now analyze the sensitivity of hyper-parameters including k_1, k_2 and m for SPAC-approx and β for white-box setting. We also illustrate the behavior of SPAC in both Fourier and spatial domains.

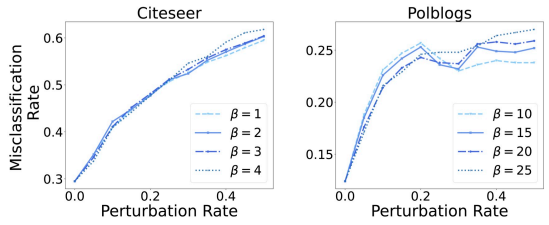


Figure 4: Sensitivity analysis on β under SPAC-CE attack.

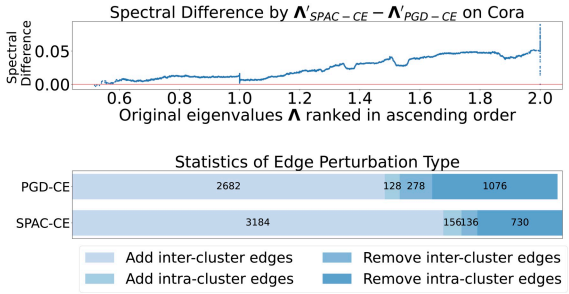


Figure 5: Analysis on spectral changes (top) and spatial edge changes (bottom) between SPAC-CE and PGD-CE.

Sensitivity of k_1, k_2 and m in SPAC-approx. For SPAC-approx, the trade-off of its attack performance and efficiency is achieved by selecting k_1 low- and k_2 high-frequency components and by approximating the eigenvalue change for m steps. Figure 6 demonstrates such trade-off under SPAC-approx poisoning attack with budget $\epsilon = 0.05$. Left side shows the misclassification rate range when using different k_1 and k_2; right side compares the misclassification rate and running time when using different approximation step m. The result suggests that the attack performance does not dramatically drop with changed parameters, and we can achieve a good balance between attack effectiveness and efficiency.

Sensitivity of hyper-parameter β in white-box setting. Figure 4 shows the performance of SPAC-CE under different settings of the coefficient parameter β. We can clearly observe that different β values lead to rather stable performance, which suggests the spectral distance term can be applied to real applications without the requirement of tedious hyper-parameter tuning.

Effect of SPAC in Fourier and spatial domain. We are interested in investigating how the changes of the graph in the Fourier domain affect its spatial structure. To serve this purpose, we compared the output of the perturbed graphs from SPAC-CE and PGD-CE on Cora under budget $\epsilon = 0.4$ in Figure 5. The top plots the difference between eigenvalues of the normalized Laplacian matrix for the graph perturbed by SPAC-CE and the graph perturbed by PGD-CE. The x-axis shows the eigenvalues of the original graph. The bottom counts the number of different types of edge perturbations, where "inter-cluster" edges are those connecting nodes with different class labels and "intra-cluster" edges connect nodes with the same class label. We observe that SPAC-CE perturbed graph in a direction leading to larger high eigenvalues and smaller low eigenvalues, compared with PGD-CE. This spectral difference in the Fourier domain is also reflected in the spatial domain: 1) more

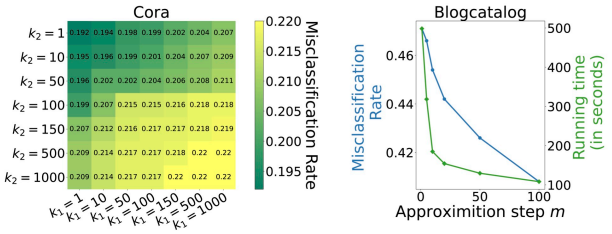
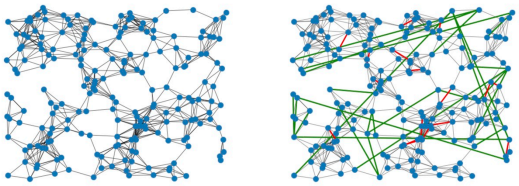


Figure 6: Sensitivity analysis on k_1, k_2 (left) and m (right) under SPAC-approx poisoning attack.



(a) Original graph (b) Perturbed graph by SPAC

Figure 7: The edge perturbation generated by SPAC on a random geometric graph with $\epsilon = 0.05$. Green denotes edges added by the attack, while red marks removed edges.

edges are added than removed; 2) specifically, more inter-cluster edges were added while fewer inter-cluster edges were removed. To intuitively demonstrate the perturbations generated by SPAC, we applied SPAC to attack the random geometric graph in Figure 1 with budget $\epsilon = 0.05$, and the perturbed graph is visualized in Figure 7. The green edges that are added by SPAC connect different node clusters, while red edges are removed within clusters. This shows that maximizing the spectral distance can modify the global connectivity of the graph: for example, SPAC-CE strengthened the connectivity between different clusters to confuse the classifier.

5 CONCLUSION

In this paper, we propose a novel graph structural attack strategy by maximizing the spectral distance between the original and the perturbed graphs. The design is motivated by the spectral perspective for understanding GCNs. An efficient approximation solution is further designed to reduce the computational complexity of spectral distance. Our experiments demonstrated the effectiveness of this new direction, and our qualitative study suggested that the proposed spectral attack tends to modify the global connectivity of a graph and enlarge the generalization gap of GCN models.

Currently, we focused on perturbing the eigenvalues of graph Laplacian, without controlling the eigenvectors. As the eigenvectors also play an important role in the spectral filters, it is important to expand our scope to manipulate eigenvectors for improved effectiveness. Applying the model-agnostic SPAC to attack a broader group of graph embedding models will also be interesting to understand the fundamental vulnerability of the graph structure.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation under grant IIS-1718216, IIS-2128019, and IIS-1553568.

REFERENCES

[1] Lada A Adamic and Natalie Glance. 2005. The political blogosphere and the 2004 US election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*. 36–43.

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

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

[4] Heng Chang, Yu Rong, Tingyang Xu, Wenbing Huang, Somayeh Sojoudi, Junzhou Huang, and Wenwu Zhu. 2021. Spectral graph attention network with fast eigen-approximation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2905–2909.

[5] Heng Chang, Yu Rong, Tingyang Xu, Wenbing Huang, Honglei Zhang, Peng Cui, Wenwu Zhu, and Junzhou Huang. 2020. A restricted black-box adversarial framework towards attacking graph embedding models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3389–3396.

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

[7] Fan RK Chung and Fan Chung Graham. 1997. *Spectral graph theory*. Number 92. American Mathematical Soc.

[8] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial attack on graph structured data. In *International conference on machine learning*. PMLR, 1115–1124.

[9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems* 29 (2016), 3844–3852.

[10] Xiaowen Dong, Dorina Thanou, Laura Toni, Michael Bronstein, and Pascal Frossard. 2020. Graph signal processing for machine learning: A review and new perspectives. *IEEE Signal Processing Magazine* 37, 6 (2020), 117–127.

[11] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. 2018. Learning structural node embeddings via diffusion wavelets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1320–1329.

[12] Chris D Godsil. 1981. On the full automorphism group of a graph. *Combinatorica* 1, 3 (1981), 243–256.

[13] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis* 30, 2 (2011), 129–150.

[14] Ming Jin, Heng Chang, Wenwu Zhu, and Somayeh Sojoudi. 2021. Power up! robust graph convolutional network against evasion attacks based on graph powering. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[15] Wei Jin, Yaxing Li, Han Xu, Yiqi Wang, Shuiwang Ji, Charu Aggarwal, and Jiliang Tang. 2021. Adversarial Attacks and Defenses on Graphs. *SIGKDD Explor. Newsl.* 22, 2 (jan 2021), 19–34. https://doi.org/10.1145/3447556.3447566

[16] Henry Kenlay, Dorina Thanou, and Xiaowen Dong. 2020. On the stability of polynomial spectral graph filters. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5350–5354.

[17] Henry Kenlay, Dorina Thanou, and Xiaowen Dong. 2021. Interpretable stability bounds for spectral graph filters. In *International conference on machine learning*. PMLR, 5388–5397.

[18] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[19] Ron Levie, Elvin Isufi, and Gitta Kutyniok. 2019. On the transferability of spectral graph filters. In *2019 13th International conference on Sampling Theory and Applications (SampTA)*. IEEE, 1–5.

[20] Jintang Li, Tao Xie, Chen Liang, Fenfang Xie, Xiangnan He, and Zibin Zheng. 2021. Adversarial attack on large scale graph. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[21] Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. 2020. Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149* (2020).

[22] Jiaqi Ma, Shuangrui Ding, and Qiaozhu Mei. 2020. Towards More Practical Adversarial Attacks on Graph Neural Networks. *Advances in Neural Information Processing Systems* 33 (2020).

[23] Yao Ma, Suhang Wang, Tyler Derr, Lingfei Wu, and Jiliang Tang. 2021. Graph Adversarial Attack via Rewiring. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD '21)*. Association for Computing Machinery, 1161–1169.

[24] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*.

[25] Jan R Magnus. 1985. On differentiating eigenvalues and eigenvectors. *Econometric theory* 1, 2 (1985), 179–191.

[26] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* 3, 2 (2000), 127–163.

[27] Bojan Mohar, Y Alavi, G Chartrand, and OR Oellermann. 1991. The Laplacian spectrum of graphs. *Graph theory, combinatorics, and applications* 2, 871-898 (1991), 12.

[28] Beresford N Parlett and David S Scott. 1979. The Lanczos algorithm with selective orthogonalization. *Mathematics of computation* 33, 145 (1979), 217–238.

[29] Mrigank Raman, Aaron Chan, Siddhant Agarwal, PeiFeng Wang, Hansen Wang, Sungchul Kim, Ryan Rossi, Handong Zhao, Nedim Lipka, and Xiang Ren. 2021. Learning to Deceive Knowledge Graph Augmented Models via Targeted Perturbation. In *International Conference on Learning Representations*.

[30] Arianna Rampini, Franco Pestarini, Luca Cosmo, Simone Melzi, and Emanuele Rodola. 2021. Universal spectral adversarial attacks for deformable shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3216–3226.

[31] Franz Rellich and Joseph Berkowitz. 1969. *Perturbation theory of eigenvalue problems*. CRC Press.

[32] Lynn C Rogers. 1970. Derivatives of eigenvalues and eigenvectors. *AIAA journal* 8, 5 (1970), 943–944.

[33] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2020. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *International Conference on Learning Representations*.

[34] Ryan Rossi and Nesreen Ahmed. 2015. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 29.

[35] Aliaksei Sandryhaila and José MF Moura. 2013. Discrete signal processing on graphs: Graph Fourier transform. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 6167–6170.

[36] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*. Springer, 593–607.

[37] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[38] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. 2018. Constructing unrestricted adversarial examples with generative models. *Advances in Neural Information Processing Systems* 31 (2018), 8312–8323.

[39] Gilbert W Stewart. 1990. Matrix perturbation theory. (1990).

[40] Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. 2020. *Adversarial Attacks on Graph Neural Networks via Node Injections: A Hierarchical Reinforcement Learning Approach*. Association for Computing Machinery, New York, NY, USA, 673–683.

[41] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations* (2018).

[42] Saurabh Verma and Zhi-Li Zhang. 2019. Stability and generalization of graph convolutional neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1539–1548.

[43] Binghui Wang and Neil Zhenqiang Gong. 2019. Attacking graph-based classification via manipulating the graph structure. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2023–2040.

[44] Binghui Wang, Tianxiang Zhou, Minhua Lin, Pan Zhou, Ang Li, Meng Pang, Cai Fu, Hai Li, and Yiran Chen. 2020. Evasion Attacks to Graph Neural Networks via Influence Function. *arXiv preprint arXiv:2009.00203* (2020).

[45] Marcin Waniek, Tomasz P Michalak, Michael J Wooldridge, and Talal Rahwan. 2018. Hiding individuals and communities in a social network. *Nature Human Behaviour* 2, 2 (2018), 139–147.

[46] Kilian Q Weinberger, Fei Sha, Qihui Zhu, and Lawrence K Saul. 2007. Graph Laplacian regularization for large-scale semidefinite programming. In *Advances in neural information processing systems*. 1489–1496.

[47] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. 2019. Adversarial Examples for Graph Data: Deep Insights into Attack and Defense. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (Macao, China) *(IJCAI'19)*. AAAI Press, 4816–4823.

[48] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. 2019. Topology Attack and Defense for Graph Neural Networks: An Optimization Perspective. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

[49] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2847–2856.

[50] Daniel Zügner and Stephan Günnemann. 2019. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *International Conference on Learning Representations (ICLR)*.

6 APPENDIX

We list the detailed gradient calculation of the spectral distance term with eigen-decomposition, the proof of Theorem 1, the attack objectives for different white-box variants of SPAC and the hyper-parameter setup.

6.1 Gradient of the Spectral Distance

Recall that we obtain the following form via the chain rule:

$$\frac{\partial \mathcal{L}_{\text{SPAC}}}{\partial \Delta_{ij}} = \sum_{k=1}^{n} \frac{\partial \mathcal{L}_{\text{SPAC}}}{\partial \lambda'_k} \sum_{p=1}^{n}\sum_{q=1}^{n} \frac{\partial \lambda'_k}{\partial \mathbf{L}'_{pq}} \frac{\partial \mathbf{L}'_{pq}}{\partial \Delta_{ij}}$$

Here is the detailed calculation of each component:

$$\frac{\partial \mathcal{L}_{\text{SPAC}}}{\partial \lambda'_k} = \frac{\lambda'_k - \lambda_k}{\|\mathbf{\Lambda} - \mathbf{\Lambda}'\|_2}$$

$$\frac{\partial \lambda'_k}{\partial \mathbf{L}'_{pq}} = \mathbf{u}'_{kp}\mathbf{u}'_{kq}$$

$$\frac{\partial \mathbf{L}'_{pq}}{\partial \Delta_{ij}} = \frac{\mathbf{C}_{ij}}{2\sqrt{d'_p d'_q}}\left(\mathbf{1}_{i=p}\frac{\mathbf{A}'_{pq}}{d'_p} + \mathbf{1}_{j=q}\frac{\mathbf{A}'_{pq}}{d'_p} - 2\mathbf{1}_{i=p,j=q}\right)$$

where d'_p is the degree on node p of the perturbed graph: $d'_p = \sum_{k=1}^{n} \mathbf{A}'_{kp}$, and similarly $d'_q = \sum_{k=1}^{n} \mathbf{A}'_{kq}$. Meanwhile, $\mathbf{A}'$ is the adjacency matrix of the perturbed graph. The indication function $\mathbf{1}_{\text{condition}}$ is 1 if the condition is true, otherwise it is 0.

6.2 Proof of Theorem 1

Proof. **Theorem 1.** For the generalized eigenvalue problem: $\mathbf{L}\mathbf{u}_i = \lambda_i \mathbf{M}\mathbf{u}_i$, if the matrix is slightly perturbed $\mathbf{L}' = \mathbf{L} + \nabla\mathbf{L}$, we aim to find the corresponding eigenvalue perturbation: $\lambda'_i = \lambda_i + \nabla\lambda_i$. From eigenvalue perturbation theory [39], we have

$$\lambda'_{\mathbf{i}} - \lambda_{\mathbf{i}} \approx \mathbf{u}_{\mathbf{i}}^{\top}(\nabla\mathbf{L} - \lambda_{\mathbf{i}}\nabla\mathbf{M})\mathbf{u}_{\mathbf{i}}$$

And for a normalized graph Laplacian $\mathbf{L}' = \mathbf{L} + \nabla\mathbf{L}$, we have $\nabla\mathbf{M} = \text{diag}(\nabla\mathbf{L} \cdot \mathbf{1}_n)$. Submitting $\nabla\mathbf{M}$ concludes the proof.

6.3 Attack Objectives for White-box Variants

Recall that SPAC can be flexibly combined with the white-box attack framework as shown in Eq. (15), which consists of a task-specific attack objective $\mathcal{L}_{\text{attack}}$ and the proposed SPAC objective $\mathcal{L}_{\text{SPAC}}$. We denote the training node set as V_0 and test node set as V_t. Different choices of $\mathcal{L}_{\text{attack}}$ result in the following variants.

SPAC-CE combines SPAC with PGD-CE [48], and maximizes the *cross-entropy* loss on the target *test set* for *evasion* attack:

$$\mathcal{L}_{\text{attack}} = \sum_{v_i \in V_t} \text{crossEntropy}(f_\theta(\mathbf{A}+\Delta, \mathbf{X})_i, y_i)$$

SPAC-C&W combines SPAC wih PGD-C&W [48], and maximizes the *negative* C&W *score* on the target test set for *evasion* attack:

$$\mathcal{L}_{\text{attack}} = -\sum_{v_i \in V_t} \max\{Z_{i,y_i} - \max_{c \neq y_i} Z_{i,c} - \kappa\}$$

where $Z_{i,c}$ denotes the prediction logit on label c, and $\kappa \geq 0$ is a confidence level of making wrong decisions. Intuitively, the C&W score evaluates how good the model can differentiate the prediction on the ground-truth label and on the label with the (second) highest likelihood. So the attack aims to confuse the model by maximizing the negative C&W score.

SPAC-Min combines SPAC and Max-Min [48], and maximizes the *cross-entropy* loss on the *training set*, while a surrogate model $f_{\theta'}$ is iteratively retrained. The perturbed graph is then used to train a victim model, and we report the classification performance of the test set on clean graph. The *poisoned* graph is generated by:

$$\mathcal{L}_{\text{attack}} = \sum_{v_i \in V_0} \text{crossEntropy}(f_{\theta'}(\mathbf{A}+\Delta, \mathbf{X})_i, y_i)$$

SPAC-Train combines SPAC with Meta-Train [50], and maximizes the *cross-entropy* loss on labeled *training nodes*, arguing that if a model has a high training error, it is likely to generalize poorly:

$$\mathcal{L}_{\text{attack}} = \sum_{v_i \in V_0} \text{crossEntropy}(f_{\theta'}(\mathbf{A}+\Delta, \mathbf{X})_i, y_i)$$

The objective is similar to SPAC-Min, but instead of retraining the surrogate model, SPAC-Train calculate *meta-gradients* on the perturbation matrix through the surrogate model.

SPAC-Self combines SPAC with Meta-Self [50], and maximizes the *cross-entropy* loss on unlabeled *test nodes* which are assigned pseudo labels predicted by the model trained on tbe clean graph:

$$\mathcal{L}_{\text{attack}} = \sum_{v_i \in V_t} \text{crossEntropy}(f_{\theta'}(\mathbf{A}+\Delta, \mathbf{X})_i, \hat{y}_i)$$

where $\hat{y}_i$ is the predicted label from the model trained on the clean graph f_θ.

6.4 Hyper-parameter Setup

For attack methods that involve projected gradient descend (e.g., SPAC/PGD-CE, SPAC/PGD-C&W, SPAC/PGD-Min, SPAC and SPAC-approx), we optimize the attack objective by gradient descent for $T = 100$ iterations; we set adaptive step size for gradient descent as $\eta = T \cdot \epsilon / \sqrt{t}$, which is related to the perturbation budget ratio ϵ, such that for each step we can use up the budget while not exceeding the budget too much. For SPAC/Meta-Train and SPAC/Meta-Self, the iteration is decided by the perturbation budget: for each step, choose the edge entry that has the largest gradient. For GF-Attack, the top-K smallest eigenvalues are selected for K-rank approximation with $K = n - 128$ following the paper's setting. For our approximation model SPAC-approx, the reported results are based on $k_1 = 128$ lowest eigenvalues and $k_2 = 64$ largest eigenvalues.