Capstone Project Phase A

Detection of Anomaly Using Graph Fairing Convolutional Networks

24-2-R-6

Table of Contents

Abstract. Anomaly detection in graph-structured data is challenging due to the data's complex relationships and diverse attributes. The current study addresses detecting anomalies in scientific literature citations using Graph Fairing Convolutional Networks (GFCNs). By graph modeling papers' citation data, with nodes represented by papers and edges represented by citations, the approach applies the power of GFCNs to identify anomalous patterns, including citation manipulation and misleading quotes in nested structural levels. The methodology integrates Node2Vec and Isolation Forest approaches for anomaly detection process initialization. In the following, the results obtained at the previous level are sequentially refined by applying GFCNs. The study will employ the Cora and PubMed datasets as a benchmark, demonstrating the abilities of the proposed method.

Keywords: Anomaly detection · Graph Neural Networks · Graph convolutional network · Jacobi method.

1 Introduction

In the evolving landscape of scientific research, ensuring the integrity and authenticity of published works is paramount. Anomalies in scientific papers, particularly those involving the misuse of quotations to align with the preferences of specific journals or influential individuals, present a significant challenge. Such practices can compromise the quality and objectivity of scientific discourse, necessitating robust detection mechanisms. this research based on the methods stated in "Detection of Anomaly Using Graph Fairing Convolutional Networks"[1] , and addressing this issue by leveraging advanced machine learning techniques to identify anomalies in scientific papers. Specifically, we focus on detecting instances where authors insert quotes strategically to increase the likelihood of publication, a practice that undermines the credibility of scientific work.

Graph Fairing Convolutional Networks (GFCNs) provide a powerful framework for this task, enabling the analysis of complex relational data within scientific texts. By modelling papers as graphs, where nodes represent textual elements and edges capture relationships between them, GFCNs can effectively detect patterns indicative of anomalous quoting behaviour. Through this approach, this study aims to enhance the transparency and reliability of scientific publications, ensuring that research integrity is maintained, and that the dissemination of knowledge remains free from undue influence.

2 Literature Review

One key challenge in anomaly detection using graph-based methods is the high variability in graph structures and node attributes across different domains and datasets. This variability can make it difficult for traditional algorithms to accurately identify anomalies in complex graph data. As a result, many research studies have focused on developing algorithms (Fig 1) that are robust to this variability, such as using deep learning techniques to learn from large datasets of graph-structured data. One of the emerging approaches in this field is the use of Graph Fairing Convolutional Networks (GFCNs), which combine the strengths of graph convolutional networks with graph fairing techniques to enhance anomaly detection performance. GFCNs have shown promise in capturing both local and global graph structures, making them particularly well-suited for detecting anomalies in complex network data [2].

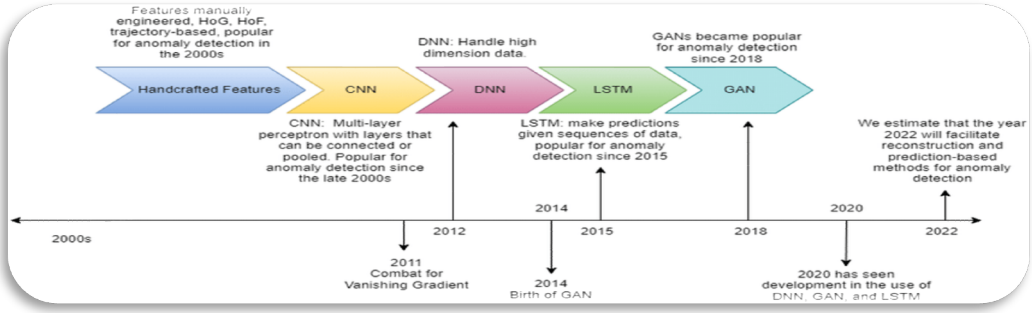


Fig. 1:Timeline for evolution of anomaly detection techniques [3]

Graph Convolutional Networks (GCNs) have become the standard for learning representations on graphs, achieving notable performance in various applications, including anomaly detection [4][5]. Ding et al. proposed an unsupervised graph anomaly detection framework using a GCN-based autoencoder, which utilizes both the topological structure and node attributes to map attribute information into a low-dimensional feature space and reconstruct the structure and attribute information from the learned representations. This approach relies on the assumption that normal instances can be reconstructed with small errors, while anomalies result in larger reconstruction errors. However, this method can be sensitive to outliers and often requires noise-free data for training. To address the challenges of label scarcity and data imbalance in graph anomaly detection, Kumagai et al. proposed semi-supervised models using GCNs for learning latent representations [5]. These models, trained to minimize the volume of a hypersphere enclosing the GCN-learned node embeddings of normal instances, also suffer from the hypersphere collapse problem. In contrast, Graph Fairing Convolutional Networks (GFCNs) leverage both graph structure and node attributes and are trained on both labelled and unlabelled data to improve performance without the hypersphere collapse issue. Graph Neural Networks (GNNs) are prone to over-smoothing, where stacking multiple layers causes node representations to become indistinguishable, leading to information loss. To tackle this, various skip connection approaches have been proposed, categorized into residual connections, initial connections, jumping connections, and dense connections. For example, JK-Net uses jumping knowledge network connections to maintain feature mappings in lower layers [6], while APPNP approximates PageRank using initial connections [7]. GCNII employs initial residual and identity mapping to mitigate over-smoothing [8]. ResGCN uses residual and dense connections, as well as dilated convolutions, to extend GCN depth [9]. In our proposed GFCN model, skip connections are applied to reuse initial node

features at each layer, combining aggregated node neighbourhood representation and initial node representation.

3 Background

3.1 Anomaly Detection

Refers to the identification of items, events, or observations that deviate significantly from the norm in a dataset. It is commonly used in various applications such as fraud detection, network security, and fault detection. The main challenge is to identify suspicious items or events even in the absence of abnormal instances. In the context of this paper, anomaly detection aims to identify graph nodes that exhibit unusual patterns compared to the majority of the nodes.

3.2 Node2Vec

Feature learning in networks aims to automate the process of deriving useful features for prediction tasks involving nodes and edges. Traditional methods often rely on hand-engineering features specific to the domain, which can be labour-intensive and may not generalize well across different datasets. Modern approaches focus on learning feature representations directly from the network structure [17].

3.2.1 Node2Vec Algorithm

Node2Vec is a prominent algorithm in this domain, designed to learn low-dimensional feature representations for nodes in a network. The key innovation of Node2Vec is its use of a flexible, biased random walk strategy to capture diverse neighbourhood structures. This approach allows the algorithm to effectively balance between local and global network properties, making it suitable for various network types and tasks.

3.2.2 Mathematical Framework

Given a network $G = (V, E)$ with nodes V and edges E, the goal of Node2Vec is to learn a mapping $f: V \to \mathbb{R}^d$ that embeds nodes into a d-dimensional space. This embedding is achieved by optimizing the likelihood of preserving network neighborhoods. Formally, for a node u, we define its neighbourhood $N_S(u)$ using a sampling strategy S.

The optimization objective is to maximize the log-probability of observing a neighbourhood $N_S(u)$ given the feature representation $f(u)$:

$$\max_f \sum_{u \in V} \log \Pr\big(N_S(u) \mid f(u) \big)$$

To make this computation feasible, we assume conditional independence and use a SoftMax function to model the likelihood of each node pair:

$$\Pr\big(n_i \mid f(u) \big) = \frac{\exp\big(f(n_i) \cdot f(u)\big)}{\sum_{v \in V} \exp\big(f(v) \cdot f(u)\big)}$$

This transforms the objective into a form suitable for optimization using stochastic gradient descent:

$$\max_f \sum_{u \in V} \left[-\log Z_u + \sum_{n_i \in N_S(u)} f(n_i) \cdot f(u) \right]$$

where $Z_u = \sum_{v \in V} \exp\big(f(u) \cdot f(v)\big)$ is the partition function, approximated via negative sampling.

3.2.3 Random Walks and Parameters

Node2Vec uses a 2nd order random walk to explore the neighbourhood of each node. The walk is guided by two parameters, p and q, which control the probability of revisiting a node and the tendency to explore further nodes, respectively. This flexibility allows the algorithm to interpolate between Breadth-First Sampling (BFS) and Depth-First Sampling (DFS), capturing both local and global network structures.

Steps Involved

1. Preprocessing: Compute transition probabilities for random walks.
2. Random Walk Simulation: Perform multiple random walks from each node to generate sequences representing neighbourhoods.
3. Optimization: Optimize the objective function using the generated sequences.

The learned node embeddings can be combined using various operators, such as the Hadamard product, to generate edge embeddings, facilitating edge-based prediction tasks.

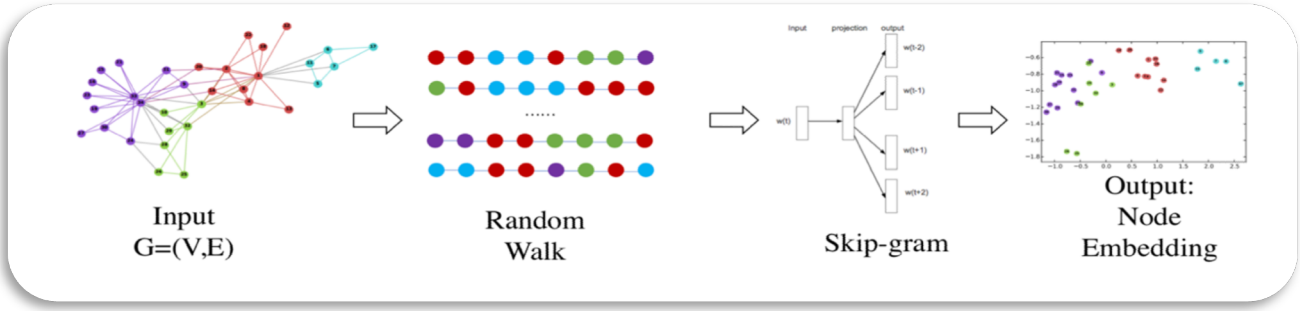


Fig2: Node2Vec Steps.

3.3 Isolation Forest (iForest)

Isolation Forest (iForest) is an innovative anomaly detection method that fundamentally differs from traditional approaches by focusing on isolating anomalies rather than profiling normal instances. This method leverages the properties of anomalies—being few in number and different from normal instances—to achieve high detection performance and efficiency [16].

3.3.1 Isolation Trees (iTrees):

iForest constructs Isolation Trees (iTrees) through recursive partitioning of data. Each iTree isolates an instance by randomly selecting an attribute and a split value, thereby partitioning the data. Anomalies, due to their distinct characteristics, are isolated closer to the root of the tree. Formally,

given a dataset X with n instances and d attributes, an iTree is built as follows:

1. Randomly select an attribute q from the d attributes.
2. Randomly select a split value p between the minimum and maximum values of q.
3. Partition the data into X_{left} where $q < p$ and X_{right} where $q \geq p$.
4. Repeat the above steps until each instance is isolated or the maximum tree height is reached.

Anomaly Score Calculation:

The anomaly detection process in iForest relies on the path length $h(x)$, the number of edges traversed from the root to the terminal node for an instance x. The anomaly score $s(x)$ is calculated using the average path length $E(h(x))$ over a collection of t iTrees. Anomalies typically have shorter path lengths. The anomaly score is defined as:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

where $c(n)$ is the average path length of unsuccessful searches in Binary Search Trees (BSTs), given by:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n}$$

with $H(i)$ as the i-th harmonic number, approximated as:

$$H(i) \approx \ln(i) + \gamma$$

and γ is the Euler-Mascheroni constant ($\gamma \approx 0.577$).

Sub-sampling:

iForest utilizes sub-sampling to construct partial models, significantly reducing computational complexity and memory requirements. The sub-sampling size ψ is crucial for the method's efficiency. Empirical studies indicate that setting ψ to values such as 256 or 512 generally yields optimal results.

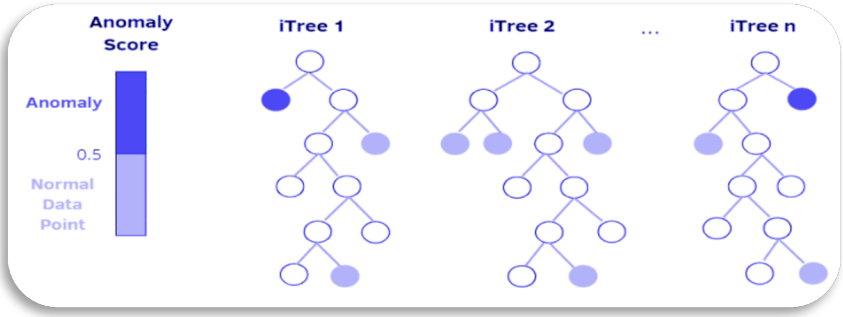


Fig3: Isolation Forest Illustration

3.4 Normalized Laplacian Matrix

A scaled version of the Laplacian matrix, designed to normalize the contributions of each node's neighbours in a graph [12]. The regular Laplacian matrix L is defined as: $L = D - A$ where D is the degree matrix and A is the adjacency matrix. The normalized Laplacian matrix modifies this to:$L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ where I is the identity matrix. The normalized version ensures that the eigenvalues are in the range [0,2], which improves the stability and convergence properties of spectral graph filtering algorithms [3]. This matrix captures the connectivity structure of the graph and is used in various graph algorithms, including spectral clustering and graph convolution.

3.5 Jacobi Method

An iterative technique for solving a matrix equation. In the context of implicit fairing, it is used to iteratively solve the implicit fairing equation. The Jacobi method splits the matrix into a diagonal and off-diagonal part and iteratively updates the solution [13]. For the implicit fairing equation $(I + sL)H = X$, the Jacobi method iterates as follows:

$$H^{(t+1)} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}H^{(t)}\Theta_s + X\widetilde{\Theta}_s$$

where $H^{(t)}$ is the solution at iteration t, Θ_s and $\widetilde{\Theta}_s$ are diagonal weight matrix, and X is the initial feature matrix [4].

3.6 Spectral Graph Filtering

A technique that uses polynomial or rational polynomial filters defined as functions of the graph Laplacian to attenuate high-frequency noise in the graph signal [14]. The filtering process involves applying a transfer function $h(L)$ to the graph signal X:

$$H = h(L)X = Uh(\Lambda)U^T X$$

where U is the matrix of eigenvectors of the Laplacian, and Λ is the diagonal matrix of its eigenvalues [5].

3.7 Graph Convolutional Network (GCN)

A neural network architecture specifically designed to operate on graph-structured data, leveraging both node features and graph topology. Unlike traditional neural networks that work with grid-like data structures (such as images or sequences), GCNs handle data that is represented as graphs, where relationships between data points are as important as the data points themselves [10].

3.7.1 Key Concepts and Equations

1. Graph Representation: A graph G is represented as $G = (V, E)$ where V is the set of nodes and E is the set of edges. The graph structure is typically encoded using an adjacency matrix A, where A_{ij} represents the edge weight between nodes i and j. The degree matrix D is a diagonal matrix where D_{ii} represents the degree of node i, which is the sum of the weights of edges connected to node i.
2. Graph Convolution Operation: The core idea of GCNs is to perform a convolution operation on the graph, which involves aggregating

feature information from a node's neighbours. The propagation rule for a single layer of a GCN can be expressed as:

$$H^{(l+1)} = \sigma\left(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

where:

- $\tilde{A} = A + I$ is the adjacency matrix with added self-loops (to include the node itself in the convolution).
- $\widetilde{D}$ is the degree matrix of $\tilde{A}$.
- $H^{(l)}$ is the feature matrix at layer l.
- $W^{(l)}$ is the trainable weight matrix at layer l.
- σ is an activation function (e.g., ReLU).

Normalization: The term $\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}$ is used to normalize the adjacency matrix. This normalization ensures that the features are properly scaled, avoiding issues with nodes having varying degrees. The normalization can be seen as a form of Laplacian smoothing, where the goal is to spread information across the graph.

Layer-Wise Propagation: GCNs typically stack multiple layers to learn higher-order feature representations. Each layer applies the convolution operation, enabling the network to capture more complex patterns. The input to the first layer is usually the node feature matrix X: $H^{(0)} = X$ Subsequent layers use the output of the previous layer as input: $H^{(l+1)} = \sigma\left(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$

Activation and Output: The activation function σ introduces non-linearity into the model, allowing it to learn more complex functions. Common choices for σ include ReLU (Rectified Linear Unit) and softmax (for classification tasks). The output layer's activation function is typically chosen based on the specific task (e.g., softmax for multi-class classification).

3.8 Graph Fairing Convolutional Network (GFCN)

An advanced neural network architecture proposed to improve the performance of Graph Convolutional Networks (GCNs) for anomaly detection on graph-structured data. The GFCN enhances GCNs by integrating skip connections inspired by the implicit fairing concept from geometry processing, enabling better propagation of information across the graph and retaining essential node features. This results in more robust node representations and improved detection of anomalies.

3.8.1 Key Concepts and Equations

1. Graph Representation: Similar to GCNs, GFCNs operate on graphs represented as $G = (V, E)$ where V is the set of nodes and E is the set of edges. The graph structure is encoded using an adjacency matrix A and a degree matrix D. Nodes have associated feature vectors that form the feature matrix X.

   Implicit Fairing: Implicit fairing is a concept from geometry processing used to smooth surfaces. In the context of graphs, it involves designing smooth graph signals to filter out high-frequency noise while retaining geometric features. The implicit fairing filter is defined by the transfer function:

   $$h_s(\lambda) = \frac{1}{1 + s\lambda}$$

   where s is a positive parameter, and λ represents the eigenvalues of the Laplacian matrix L.

2. Implicit Fairing Equation: The implicit fairing process can be formulated as solving the following sparse linear system: $(I + sL)H = X$ where L is the normalized Laplacian matrix, H is the smoothed graph signal, X is the initial feature matrix, and s is the smoothing parameter. This equation can be derived from minimizing the following objective

function: $J(H) = \frac{1}{2}|H - X|_F^2 + \frac{s}{2}\text{tr}(H^T LH)$ where $||\cdot||_F$ denotes the Frobenius norm and tr(·) denotes the trace operator.

3. Jacobi Iterative Method: The Jacobi method is used to iteratively solve the implicit fairing equation. The iteration rule is: $H^{(t+1)} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}H^{(t)}\Theta_s + X\widetilde{\Theta_s}$
   where $H^{(t)}$ is the solution at iteration t, Θ_s and $\widetilde{\Theta_s}$ are diagonal weight matrices, and X is the initial feature matrix.
4. Layer-Wise Propagation Rule: The core innovation of GFCN is its layer-wise propagation rule, which combines graph convolution with skip connections. The propagation rule for a single layer is given by:

$$H^{(l+1)} = \sigma\left(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}H^{(l)}\Theta^l + X\widetilde{\Theta}^{(l)}\right)$$

   where σ is a non-linear activation function (e.g., ReLU), $\Theta^{(l)}$ and $\widetilde{\Theta^{(l)}}$ are learnable weight matrices, and $H^{(l)}$ is the feature matrix at layer l.

5. Skip Connections: Skip connections in GFCNs enable the network to reuse the initial node features at each layer, helping to preserve essential information and improve gradient flow during training. This mitigates the over-smoothing problem where node features become indistinguishable in deep networks.
6. Activation and Output: The activation function σ introduces non-linearity into the model, allowing it to learn more complex functions. The final output layer's activation function depends on the specific task, such as using softmax for classification tasks.

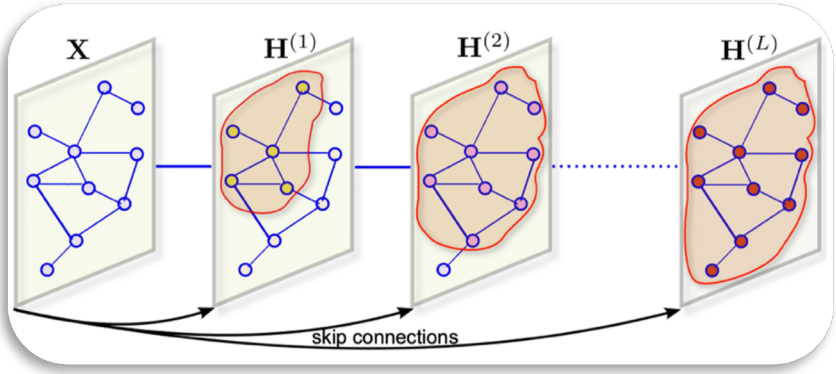


Fig. 2: Illustration of the GFCN aggregation scheme with skip connections.

3.9 Skip Connection

This is a technique used to mitigate the over-smoothing problem in deep neural networks by preserving the initial node features at each layer. In the GFCN model, skip connections help combine the aggregated node neighbourhood representation with the initial node representation [11]. Mathematically, the skip connection can be represented as:

$$H^{(l+1)} = \sigma\left(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}H^{(l)}\Theta^{l} + X\widetilde{\Theta}^{(l)}\right)$$

where X is the initial feature matrix reused at each layer ℓ. [2]

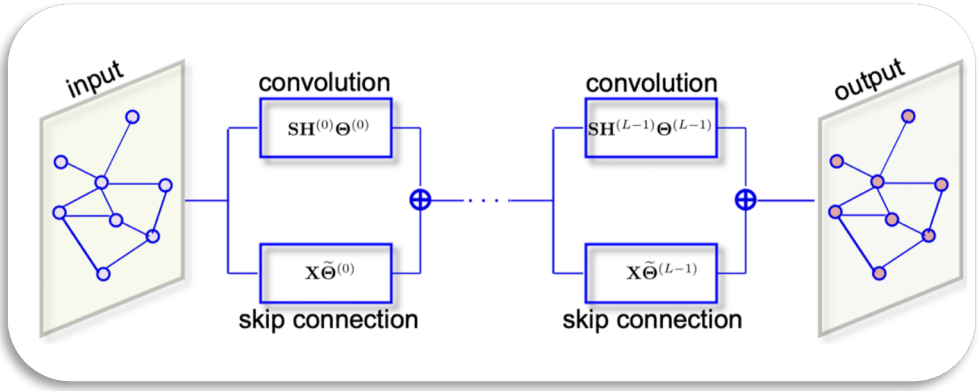


Fig. 3: Schematic layout of the proposed GFCN architecture

3.10 Implicit Fairing

Implicit Fairing refers to the process of designing and computing smooth graph signals to filter out high-frequency noise while retaining geometric features [15]. The implicit fairing filter has atransfer function:

$$h_s(\lambda) = \frac{1}{1 + s\lambda}$$

and is applied by solving the implicit fairing equation:

$$(I + sL)H = X$$

where s is a positive parameter, L is the Laplacian matrix, and X is the initial graph signal [6].

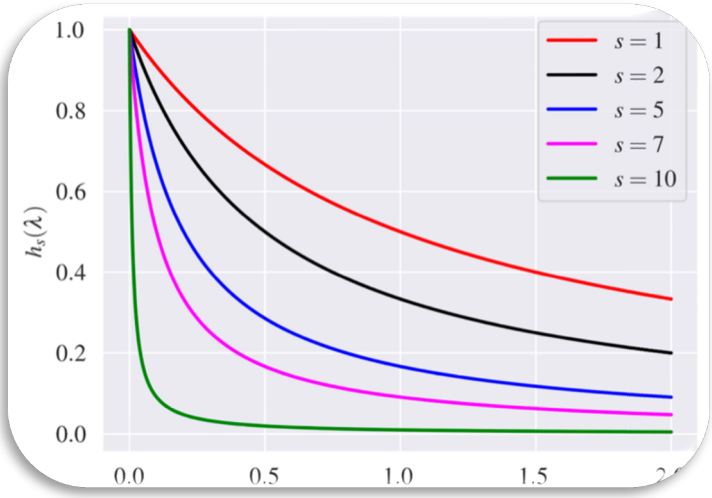


Fig. 4: Transfer function of the implicit fairing filter for various values of the scaling parameter.

3.11 Cora Dataset

The Cora dataset is a fundamental benchmark dataset extensively utilized for evaluating machine learning algorithms on graph-structured data. It plays a significant role in research areas such as network analysis, graph learning, and semi-supervised learning due to its well-defined structure and comprehensive features. The dataset is composed of nodes and edges, where each node represents a scientific paper and each edge represents a citation link between two papers. Specifically, an edge from one node to another signifies that the first paper cites the second paper. The dataset categorizes these papers into seven distinct classes, each representing a different topic within the field of machine learning. These topics include Case-Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning, and Theory. Each paper is described by a binary word vector that indicates the presence or absence of specific words, with the length of the vector corresponding to the vocabulary size of the dataset. In terms of data statistics, the Cora dataset comprises 2,708 nodes (papers) and 5,429 edges (citations). Each paper is characterized by a binary word vector of length 1,433, representing the presence or absence of certain words. The seven classes within the dataset provide a diverse range of topics for analysis and classification tasks. The citation graph in the Cora dataset is directed, meaning that edges have a direction from the citing paper to the cited paper. This directed nature of the graph captures the flow of information and influences among the papers, making it a realistic representation of academic

citations. The Cora dataset is widely used for various machine learning tasks, including node classification, link prediction, and clustering. Node classification involves predicting the topic of a paper based on its feature vector and citation links. Link prediction aims to predict the existence of a citation link between two papers. Clustering groups papers into clusters based on their features and citation patterns, revealing underlying structures and relationships within the dataset.

4 Engineering Process

The engineering process for our anomaly detection framework using Graph Fairing Convolutional Networks (GFCNs) involves several key steps: graph creation, initialization using Node2Vec and Isolation Forest, outlier detection, removal of outlier nodes and edges, and iteration for refinement. Here, we detail each step comprehensively.

4.1 Graph Creation

The initial step involves constructing a graph from the given dataset. Each node in the graph represents an individual paper, and edges represent the citation relationships between these papers. The steps involved in graph creation are as follows:

1. Data Preprocessing: Clean and preprocess the raw data to ensure it is suitable for graph construction. This includes handling missing values, normalizing the data, and preparing the features.
2. Node and Edge Definition: Define the criteria for nodes and edges based on the dataset. This involves determining the attributes of nodes (papers) and the nature of connections (citations) between them.
3. Graph Construction: Use graph theory techniques to construct the initial graph, ensuring all nodes and edges are appropriately defined and connected.

4.2 Initialization Step

Once the graph is constructed, the next step is to initialize the node embeddings and identify potential anomalies using the following methods:

4.2.1 Node2Vec Transformation

Node2Vec is employed to transform the graph into a set of feature vectors. This algorithm performs biased random walks on the graph to generate sequences of nodes, capturing both local and global structural information. These sequences are then used to learn low-dimensional embeddings for each node.

1. Graph Representation: Given a graph $G = (V, E)$ with nodes V and edges E, Node2Vec learns an embedding $f: V \rightarrow \mathbb{R}^d$ by maximizing the likelihood of preserving network neighborhoods.
2. Optimization Objective: For a node u, the objective is to maximize the log-probability of observing its neighborhood $N_S(u)$, sampled using a biased random walk strategy.
3. Random Walks: The biased random walks are controlled by two parameters, p and q, which determine the likelihood of revisiting a node or exploring new nodes, balancing between Breadth-First Sampling (BFS) and Depth-First Sampling (DFS).

4.2.2 Isolation Forest for Anomaly Detection

The embeddings generated by Node2Vec are then fed into an Isolation Forest (iForest) to identify anomalous nodes.

Isolation Trees (iTrees): iForest constructs multiple Isolation Trees by recursively partitioning the data. Each tree isolates an instance by randomly selecting an attribute and a split value. Anomalies, being few and different, tend to be isolated closer to the root of the tree.

Anomaly Scoring: The anomaly score for each instance is computed based on the path length from the root to the terminal node. Shorter path lengths correspond to higher anomaly scores. The formula for the anomaly score $s(x)$ of an instance x is:

$$s(x) = 2^{-\frac{E(h(x))}{c(n)}}$$

where $E(h(x))$ is the average path length of x in the forest and $c(n)$ is the average path length of unsuccessful searches in Binary Search Trees (BSTs).

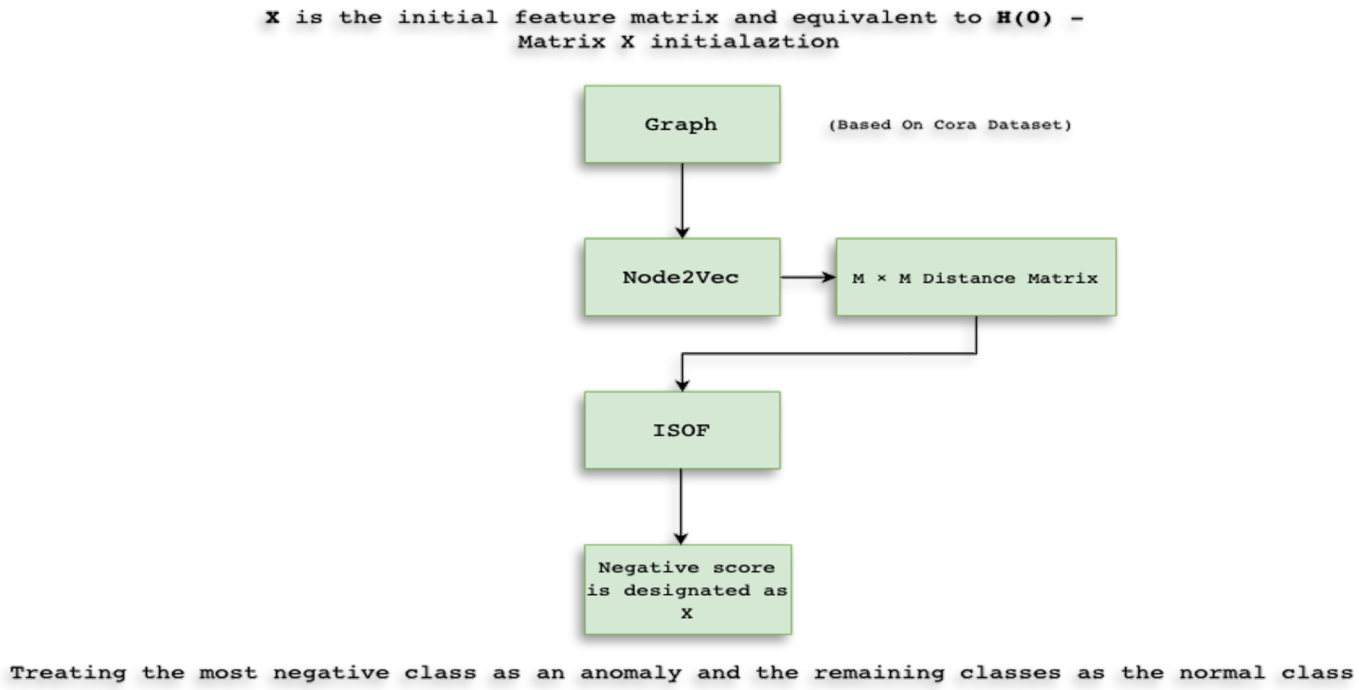


Fig. 5: Initial Process

4.3 Finding Outliers

After the initialization step, the next phase involves detecting outlier nodes and edges that may represent anomalies using the GFCN framework.

1. Feature Extraction: Extract features from the graph using GFCNs. These features help identify patterns and irregularities within the graph.

2. Outlier Detection: Implement algorithms to detect outliers based on the extracted features. Techniques such as clustering, density estimation, or deviation analysis are used to identify nodes and edges that significantly differ from the rest of the graph.

4.4 Removing Outlier Nodes and Edges

Once the outliers are detected, the identified anomalous nodes and edges are removed from the graph to refine its structure and improve the accuracy of the anomaly detection process.

1. Outlier Analysis: Analyze the detected outliers to understand their impact on the graph.
2. Removal Process: Implement a method to remove the identified outlier nodes and edges, which may involve adjusting the graph structure and recalculating the connections between the remaining nodes.

4.5 Repeating the Process

The final step is to iterate the entire process to ensure thorough analysis and detection of all significant anomalies. This iterative approach helps refine the graph and enhances the robustness and accuracy of the anomaly detection system.

1. Iteration: Repeat the steps of graph construction, initialization, outlier detection, and removal multiple times.
2. Refinement: In each iteration, refine the graph and the detection process based on the results of the previous iteration.
3. Validation: Validate the final graph to ensure it is free of significant anomalies and accurately represents the underlying data.

This process ensures that our GFCN-based framework effectively detects and isolates anomalies within the Cora dataset, thereby improving the integrity and reliability of scientific publications.

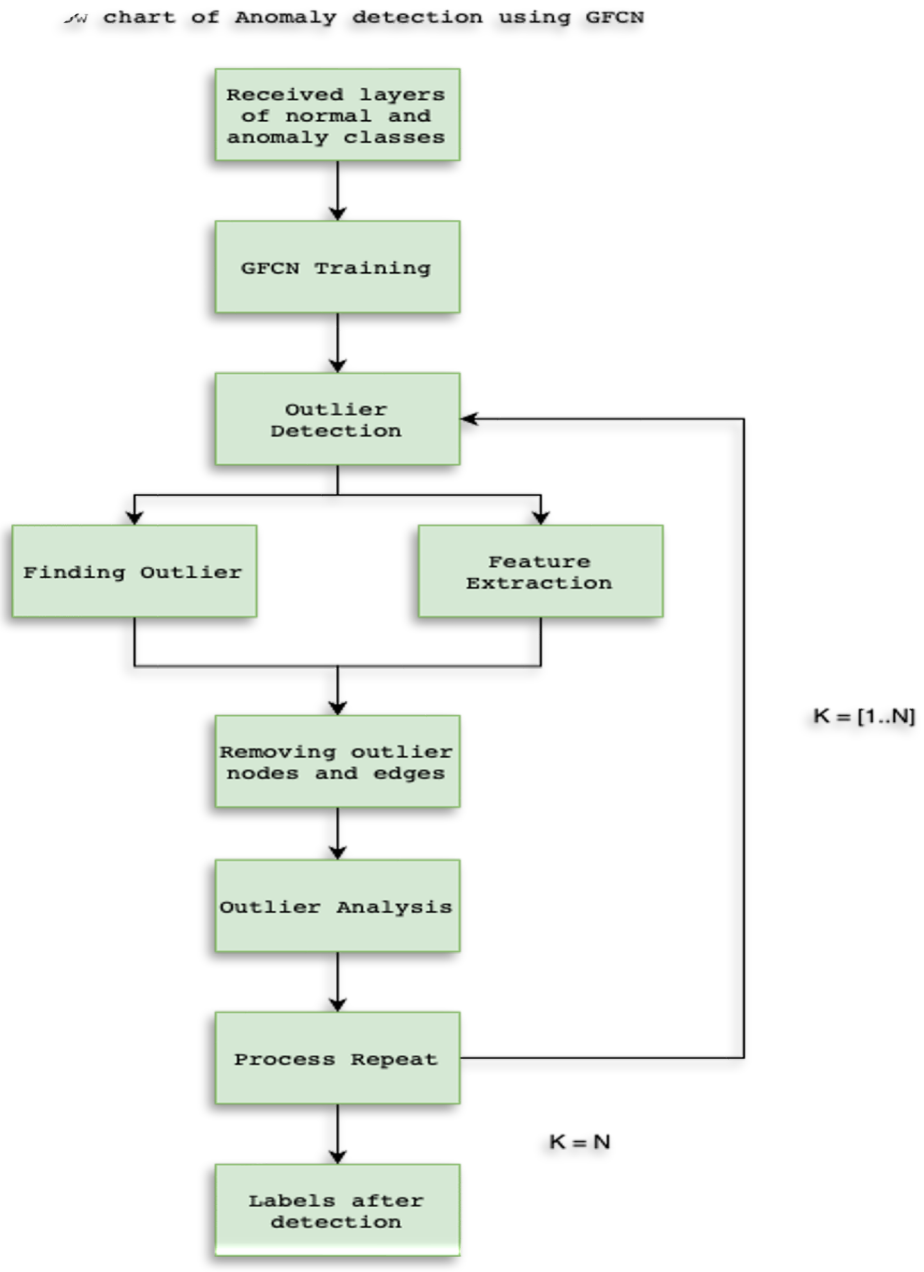
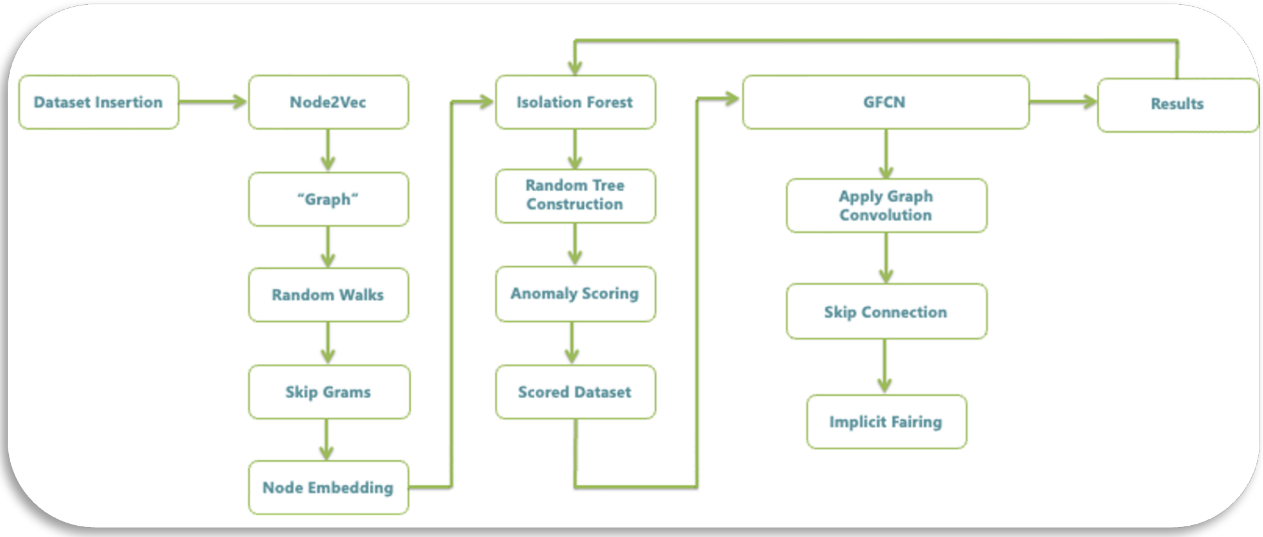
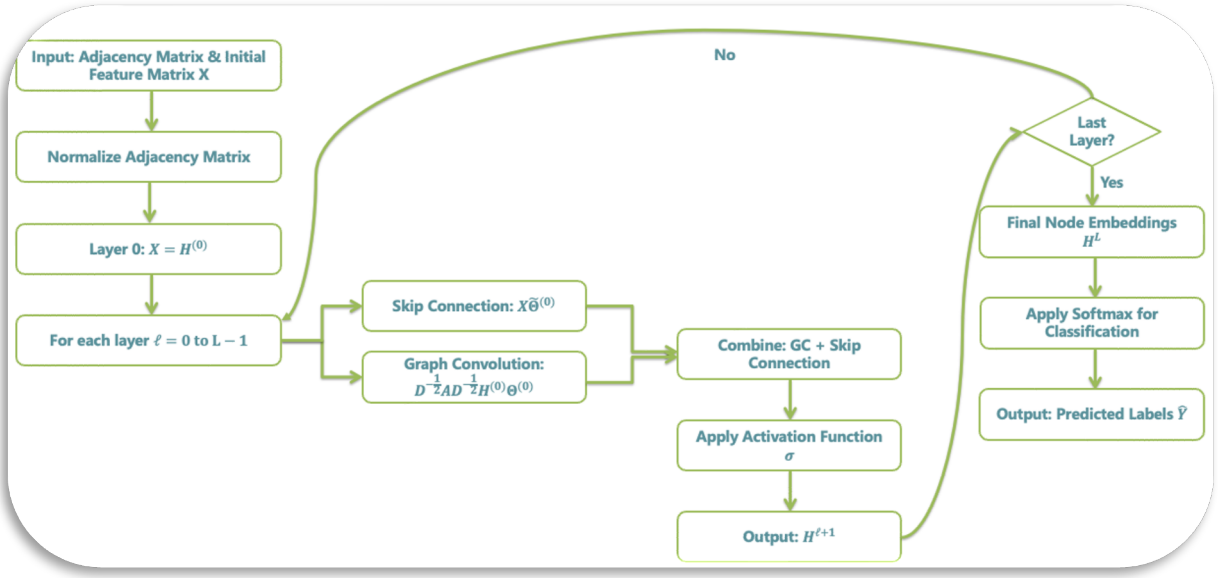


Fig. 6: GFCN Process Steps.

5 Model's Process



6 Elaborated diagram of GFCN



7 Expected Achievements

Our primary goal is to achieve high-accuracy anomaly detection in scientific articles using Graph Fairing Convolutional Networks (GFCNs). We aim to detect anomalies at various levels, including document structure, citation patterns, and content consistency. By applying GFCNs to the complex network of scientific literature, we expect to identify unusual publication patterns, aberrant citation behaviours, and potentially fraudulent or low-quality research outputs. Our approach will involve examining common

spectral filters on graphs and proposing a graph convolutional network with skip connections, utilizing the concept of implicit fairing on graphs. We will analyse the main components of this architecture and its computational complexity. Additionally, we plan to introduce an anomaly scoring function based on the weighted cross-entropy between ground-truth labels of graph test nodes and the model's predicted probabilities. Our target is to achieve a detection accuracy above 80%, significantly improving upon current methods. We anticipate that our GFCN-based approach will be particularly effective in capturing both local and global anomalies within the scientific literature graph, providing a more nuanced and context-aware detection system. This research has the potential to enhance the integrity of scientific publications and assist in maintaining the quality of academic databases.

8 Evaluation / Verification Plan

In this section, we test the core functionalities and algorithms. By systematically testing components such as Node2Vec, GFCN, and Isolation Forest, we ensure that the system performs reliably, accurately, and efficiently. The following test cases cover key scenarios to validate the system's ability to generate embeddings, classify nodes, detect anomalies, and handle various data challenges.

Case	Test Case	Expected Result
1	Run Node2Vec on a sample graph dataset	Embeddings for nodes are generated and stored successfully.
2	Run Node2Vec on a sample graph dataset with q = 0.25 and p = 0.5	Node embeddings are generated and stored for each node in a low-dimensional space with a meaningful clustering and clear separation of anomalies
3	Detection against ground truth	precision > 0.75, recall > 0.70

4	A large citation graph dataset with over 5000 nodes and edges	The entire pipeline (Node2Vec, Isolation Forest, and GFCN) runs efficiently within an expected time frame, scaling with the dataset size.
5	Detect anomalies using Isolation Forest	Anomalies are identified and flagged in the dataset.
6	Apply GFCN on a training dataset	GFCN model trains and outputs node classifications.
7	Test scalability on a large graph dataset	Algorithms run efficiently and complete within expected time.
8	Validate anomaly detection precision	Precision and recall metrics meet specified thresholds.

Bibliography

[1] Mesgaran, M., & Hamza, A. B. (2020). “Graph fairing convolutional networks for anomaly detection.”

[2] Zhang, Y., Li, X., & Wang, L. (2021). "Graph Fairing Convolutional Networks for Anomaly Detection in Attributed Graphs." In Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), pp. 2164-2173

[3] Anomaly detection using edge computing in video surveillance system − June 2022 − Research Gate

[4] K. Ding, J. Li, R. Bhanushali, and H. Liu, “Deep anomaly detection on attributed networks,” in Proc. SIAM Interna- tional Conference on Data Mining, pp. 594–602, 2019.

[5] A. Kumagai, T. Iwata, and Y. Fujiwara, “Semi-supervised anomaly detection on attributed graphs,” in Proc. Interna- tional Joint Conference on Neural Networks, 2021.

[6] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka, “Representation learning on graphs with jumping knowledge networks,” in Proc. International Conference on Machine Learning, 2018.

[7] J. Gasteiger, A. Bojchevski, and S. Günnemann, “Combining neural networks with personalized pagerank for classification on graphs,” in Proc. International Conference on Learning Representations, 2019.

[8] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, “Simple and deep graph convolutional networks,” in Proc. International Conference on Machine Learning, pp. 1725–1735, 2020.

[9] G. Li, M. Muller, A. Thabet, and B. Ghanem, “DeepGCNs: Can GCNs go as deep as CNNs?,” in Proc. IEEE International Conference on Computer Vision, pp. 9267–9276, 2019.

[10] Kipf, T. N., & Welling, M. (2016). "Semi-Supervised Classification with

Graph Convolutional Networks."
[11] He, K., Zhang, X., Ren, S., & Sun, J. (2016). "Deep Residual Learning for Image Recognition."
[12] Chung, F. R. K. (1997). "Spectral Graph Theory."
[13] Golub, G. H., & Van Loan, C. F. (1996). "Matrix Computations."
[14] Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A.,
& Vandergheynst, P. (2013). "The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains."
[15] Desbrun, M., Meyer, M., Schröder, P., & Barr, A. H. (1999). "Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow."
[16] Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (pp. 413-422).
[17] Grover, A., & Leskovec, J. (2016). node2vec: Scalable Feature Learning for Networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 855-864). ACM.