

Article

Robust Graph Neural-Network-Based Encoder for Node and Edge Deep Anomaly Detection on Attributed Networks [†]

G. Victor Daniel ^{1,*} , Kandasamy Chandrasekaran ¹, Venkatesan Meenakshi ²  and Prabhavathy Paneer ³ 

¹ Department of Computer Science and Engineering, National Institute of Technology Karnataka, Mangalore 575025, India; kch@nitk.edu.in

² Department of Computer Science and Engineering, National Institute of Technology Puducherry, Puducherry 609609, India; venkatesan.msundaram@nitpy.ac.in

³ School of Information Technology and Engineering, Vellore Institute of Technology Vellore, Vellore Campus, Vellore 632014, India; pprabhavathy@vit.ac.in

* Correspondence: victor.197541@nitk.edu.in

[†] This research is an extended conference paper presented at the 11th International Conference on Cloud Computing, Data Science & Engineering, Noida, India, 28–29 January 2021; doi:10.1109/Confluence51648.2021.

Abstract: The task of identifying anomalous users on attributed social networks requires the detection of users whose profile attributes and network structure significantly differ from those of the majority of the reference profiles. GNN-based models are well-suited for addressing the challenge of integrating network structure and node attributes into the learning process because they can efficiently incorporate demographic data, activity patterns, and other relevant information. Aggregate operations, such as sum or mean pooling, are utilized by Graph Neural Networks (GNNs) to combine the representations of neighboring nodes within a graph. However, these aggregate operations can cause problems in detecting anomalous nodes. There are two main issues to consider when utilizing aggregate operations in GNNs. Firstly, the presence of anomalous neighboring nodes may affect the representation of normal nodes, leading to false positives. Secondly, anomalous nodes may be overlooked as their representation is flattened during the aggregate operation, leading to false negatives. The proposed approach, AnomEn, is a robust graph neural network developed for anomaly detection. It addresses the challenges of false positives and false negatives using a weighted aggregate mechanism. This mechanism is designed to differentiate between a node's own features and the features of its neighbors by placing greater emphasis on a node's own features and less emphasis on its neighbors' features. The system can preserve the node's original characteristics, whether the node is normal or anomalous. This work proposes not only a robust graph neural network, namely, AnomEn, but also specific anomaly detection structures for nodes and edges. The proposed AnomEn method serves as the encoder in the node and edge anomaly detection architectures and was tested on multiple datasets. Experiments were conducted to validate the effectiveness of the proposed method as a graph neural network encoder. The findings demonstrated the robustness of the proposed method in detecting anomalies. The proposed method outperforms other existing methods in node anomaly detection tasks by 5.63% and edge anomaly detection tasks by 7.87%.

Keywords: anomaly detection; attributed network embedding; social networks; graph neural-network-based encoder; graph neural networks



Citation: Daniel, G.V.; Chandrasekaran, K.; Meenakshi, V.; Paneer, P. Robust Graph Neural-Network-Based Encoder for Node and Edge Deep Anomaly Detection on Attributed Networks. *Electronics* **2023**, *12*, 1501. <https://doi.org/10.3390/electronics12061501>

Academic Editor: Ping-Feng Pai

Received: 9 February 2023

Revised: 14 March 2023

Accepted: 16 March 2023

Published: 22 March 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Attributed networks can be utilized to depict social connections, where the graph nodes denote individuals and the edges between them signify their relationships. Attributed networks provide a wealth of attributes in addition to network topology. Anomaly detection is the process of identifying patterns or data points that do not conform to the

expected or normal behavior of a system or dataset. It involves analyzing data to find unusual patterns, behaviors, or events that could indicate a potential problem or opportunity for further investigation. Anomaly detection in attributed networks aims to identify users whose characteristics deviate significantly from the norm, and has applications in areas such as detecting social spammers, bots, and fake accounts [1].

In a social network, the norm refers to the typical behavior of users with a particular set of attributes, such as the frequency of posting updates, the number of followers, or the topics they usually discuss. Anomaly detection techniques aim to identify users whose behavior deviates significantly from this norm, indicating that they may be engaged in suspicious or malicious activities, such as spamming, phishing, or spreading false information.

The study of methods for social network anomaly detection is presently gaining popularity. Graphs are classified as non-Euclidean data because they neither follow a grid pattern nor do they obey Euclidean geometry [2]. Unlike traditional data, which are often represented in a tabular format, graphs can have complex structures, and relationships between nodes can be non-linear and non-transitive. As a result, traditional data analysis techniques may not be suitable for analyzing social network data. Anomaly detection is one such technique that involves identifying unusual patterns or behaviors in a network that deviate from normal or expected behavior.

2. Related Work

Recently, a number of methods for using machine learning to find anomalies in graphs have been suggested [3–5]. The use of machine-learning-based algorithms in anomaly detection on graphs can be challenging. It often involves manual feature selection, which is unsuited for large networks with many features [6,7]. The rich and essential information that resides in graphs is beyond what can be grasped by analyzing individual nodes or structure information alone [7,8]. When examining isolated data points, it may be difficult to see complicated relationships or patterns that exist in a graph or network. It might be possible to gain important insights or reach conclusions by looking at the graph's structure as a whole that cannot be obtained by looking at individual components separately. The use of matrix decomposition and matrix factorization techniques has been used for anomaly detection [9,10], but with large graphs they do not perform well. The detection of graph anomalies has recently been studied using deep learning architectures [11–13].

In order to expand CNN techniques to non-Euclidean data, authors M. Henaff et al. [14] and Niepert et al. [15] suggested models for applying convolutional neural networks to graph data. A model put forth by Niepert et al. places nodes in relation to structural factors such as betweenness centrality and the number of neighbors. It then gives a set of adjacent nodes with a fixed size and a sequence of nodes with a fixed length. CNNs can be applied to the graph using the model, which assigns node numbers using graph labeling methods. However, as pointed out in the citation Zhang et al., 2019 [16], this method heavily depends on the structure of the graph and might need to generalize to a wider variety of applications.

A type of neural network referred to as a graph neural network (GNN) is designed to operate on graph-structured data, which represents a group of entities (known as "nodes") and their interconnections (known as "edges") using mathematical techniques.

To create node embeddings for node classification, Max Kipf introduced the Graph Convolutional Network (GCN), a graph-based neural network [17]. It creates a node's representation in the network by gathering its neighbors' attributes, and creates node embeddings by presenting each node as a combination of its neighbors' features. Kaize Ding (2019) [18] proposed a methodology that makes use of autoencoders for anomaly detection in light of this work. A total of two GCN layers are used in this work's autoencoder architecture for anomaly detection.

In their study, Zhang et al., 2022 [19] introduced a novel fraud detection system called eFraudCom, which utilizes a competitive graph neural network (CGNN) to identify fraudulent activities on an e-commerce platform. The CGNN system is based on graph

convolutional networks (GCN) and graph autoencoders (GAE). While this approach may be effective for detecting fraud in e-Commerce applications, the approach is not robust to the presence of anomalous nodes, which can affect the system's accuracy in detecting fraud or other anomalies.

2.1. Limitation of the Existing Methods

Graph neural network (GNN)-based approaches have shown promise for anomaly detection in graphs. However, these methods face two primary challenges.

Firstly, the latent representation of nodes in the graph can be significantly impacted by the presence of anomalous nodes in their immediate neighborhood. This effect can lead to the incorrect classification of normal nodes as anomalous, compromising the accuracy of the anomaly detection approach. This issue arises due to the nature of GNNs, where the representation of a node is determined by aggregating the representations of its neighboring nodes. As a result, the presence of anomalous nodes can significantly influence the latent representation of a node, leading to false positives.

Secondly, anomalous nodes may not be detected by GNN-based approaches since their representations are normalized by the aggregate of genuine nodes in their neighborhood. This means that anomalous nodes may be masked by the presence of numerous normal nodes in their immediate vicinity. As a result, such nodes may not be effectively flagged as anomalous, leading to a failure of the anomaly detection approach.

2.2. Contributions of This Paper

The extended version of the paper presents a novel architecture for edge anomaly detection that integrates the encoder proposed in the original conference paper [20]. While the original paper focused solely on node anomaly detection, the extended version includes an evaluation of the proposed encoder on edge anomaly detection. The results indicate that the proposed encoder outperforms current GNN encoders for detecting edge anomalies. Additionally, the proposed edge anomaly detection architecture, when used in combination with the proposed encoder, outperforms existing edge anomaly detection methods.

The proposed model, Anomaly Encoder (AnomEn), addresses the issue of accurately reflecting a node's inherent characteristics by balancing the weight assigned to a node's own features and its neighbors' features. The proposed AnomEn method differs from DOMINANT [18] in that it balances the weight between a node's own features and its neighbors' features during the aggregation process to obtain the latent representation.

In summary, the primary contributions of this paper are:

- Proposal of a new network-embedding technique called Anomaly Encoder (AnomEn) for anomaly detection. The method weights the self-features and neighborhood features of the node to create a representation of the node. This representation is then used as an encoder in an anomaly detection framework that aims to identify anomalies in the network.
- Proposal of a framework for attributed networks based node anomaly identification. The system employs the proposed encoder, AnomEn, to generate latent representations of nodes, and two decoders: a structure reconstruction decoder and an attribute reconstruction decoder. The framework aims to identify node anomalies in attributed networks.
- Introduction of an edge anomaly detection architecture built on an auto-encoder that makes use of the AnomEn encoder presented in this work. The architecture aims to detect edge anomalies in attributed networks.

3. Notations and Problem Statement

This section provides a formal definition of frequently used notations and the problem under study. Table 1 summarizes the notations used in the paper.

Table 1. Summary Statistics for the Datasets.

Matrix/Dataset	Twitter	Enron	Amazon
Adjacency matrix	2050 × 2050	13,533 × 13,533	1418 × 1418
Feature matrix	2050 × 15	13,533 × 18	1418 × 21
Ground truth	2050 × 1	13,533 × 1	1418 × 1
Anomalies	100	700	70

Consider the network with node attributes $G = (V, E)$, where V is the set of nodes and E is the collection of edges.

Problem: Given an input of an attributed undirected network ‘ G ’, the goal is to identify all anomalous nodes that deviate significantly in terms of their structure and attribute information from the majority of the nodes.

4. Methodology

The Anomaly Encoder (AnomEn) is a proposed graph neural network that serves as an encoder that takes in a high-dimensional adjacency matrix and feature matrix and outputs a low-dimensional vector representation. The generated vector representations of the nodes can be used as input to machine learning models that can classify nodes as normal or anomalous.

The low-dimensional vector produced by AnomEn can be used for both node anomaly detection and edge anomaly detection tasks. The proposed architecture for these tasks utilizes the AnomEn encoder to extract relevant information from the input data and perform anomaly detection.

4.1. Proposed Graph Neural Network: AnomEn

The AnomEn encoder takes inspiration from the GCN model. The GCN model is a graph neural network that uses convolutional operations to generate a latent representation of the graph data. To form a representation of the central node, this process aggregates the features of the neighboring nodes. The GCN model has a weakness in that it may be impacted by anomalous neighboring nodes. The representation of normal nodes may be affected, leading to false positive detections. Additionally, the representation of anomalous nodes may be flattened by the aggregate operation, which can lead to false negatives and cause them to go undetected. These limitations highlight the need for a more robust and effective approach to graph representation learning, such as the AnomEn encoder, which balances the contribution of the node’s self features and attributes of its neighbors to create a more accurate representation of the graph data as shown in Algorithm 1.

The proposed encoder is inspired by the Graph Convolutional Network (GCN) model [21], which generates latent representation by aggregating the features of neighboring nodes. GCN is not immune to the existence of anomalous nodes, though. So, as illustrated in Algorithm 1, a weighted sum of a node’s self-features and those of its neighbours is proposed.

As illustrated in the algorithm, the feature matrix of each node forms its initial representation.

$$h_v^0 \leftarrow X_v, \forall v \in N(v)$$

Neighbors send messages to each other. Every node receives messages from its neighbors as shown below.

$$h_{N(v)}^k \leftarrow \text{AGGREGATE}(h_u^{k-1}, \forall u \in N(v))$$

A node then combines its features with its neighbors' features and gives greater weight than its neighbors' features as shown below. Here $N(v)$ represents the set of neighbors of vertex v in a graph.

$$h_v^k \leftarrow \sigma(W^k \cdot \text{COMBINE}(\beta * h_v^{k-1}, (\beta - 1) * h_{N(v)}^k))$$

In this method, each node has a latent representation that reflects its characteristics and is less affected by anomalous neighbors.

Algorithm 1: Weighted Neighbourhood Aggregation Procedure

Input : Adjacency matrix A , Feature matrix X

Output: Vector representations $Z_u, u \in V$

$h_v^0 \leftarrow X_v, \forall v \in N(v)$

foreach K **do**

foreach $v \in v$ **do**

$h_{N(v)}^k \leftarrow \text{AGGREGATE}(h_u^{k-1}, \forall u \in N(v))$

$h_v^k \leftarrow \sigma(W^k \cdot \text{COMBINE}(\beta * h_v^{k-1}, (\beta - 1) * h_{N(v)}^k))$

end

$h_v^k \leftarrow \text{NORMALIZE}(h_v^k, \forall v \in v)$

end

4.2. Node Anomaly Detection Task

A viable method to identify anomalies is to make use of the discrepancy between the input and reconstructed data, as suggested by methods [17,21,22]. The related node is considered anomalous when the difference between the input and reconstructed data exceeds a predetermined threshold. Nodes that do not follow the patterns set by other nodes cannot properly reconstruct the original data. A deep autoencoder creates the latent representation of the input and uses it to reconstruct the original data in an unsupervised way.

The proposed encoder, AnomEn, is utilized to introduce a new architecture for the purpose of detecting node anomalies. This architecture is called NodeAnomEn, and it is an end-to-end framework for learning joint representations to detect anomalies in attributed networks, as depicted in Figure 1. The approach is based on an unsupervised learning mechanism that utilizes an autoencoder. Autoencoders consist of an encoder and a decoder and are primarily used for reconstructing input data. In this case, the encoder takes the network structure and attribute information as input and generates a node embedding. The decoder then uses this node embedding to reconstruct the network structure and attribute information. Any discrepancies between the reconstructed information and the input network are considered anomalies. The NodeAnomEn structure is composed of three elements. Firstly, the AnomEn encoder serves as a starting point, building a reliable latent representation of nodes using network structure and attribute data [20]. Secondly, an attribute reconstruction decoder reconstructs the node attribute data. Thirdly, a structural decoder reconstructs the network structure data. By calculating the difference between the reconstructed and actual data, anomalies in a dataset can be identified.

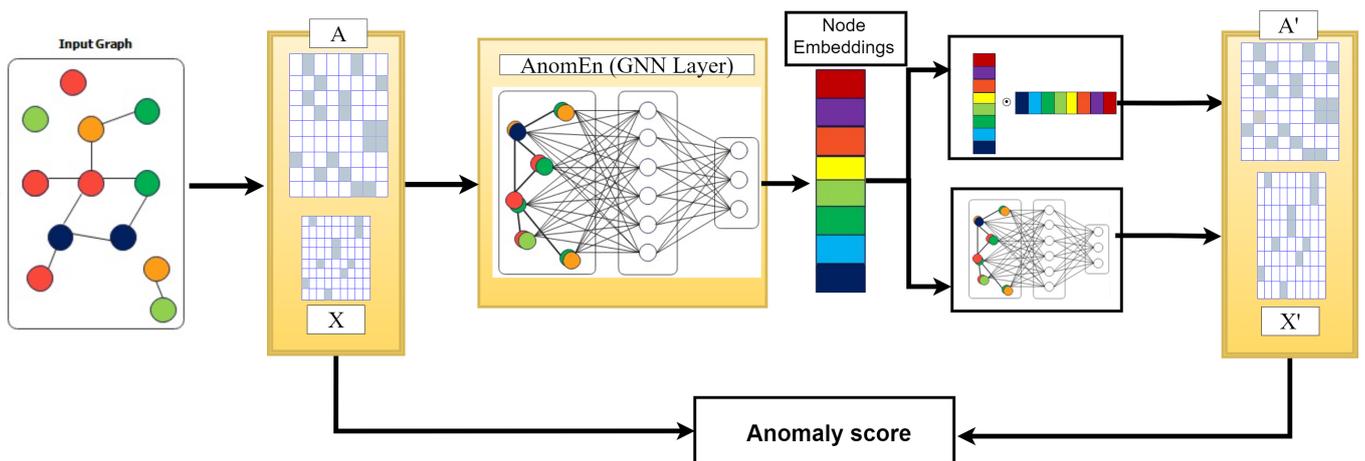


Figure 1. The Proposed Framework for Node Anomaly Detection (NodeAnomEn) in Attributed Networks.

4.2.1. Network Encoder

In total, two convolutional layers are used to map the input to node embeddings, which can be formulated as follows:

$$\widetilde{Z}^E = Relu(D^{-1/2}(A(1 - \beta) + I\beta)D^{-1/2}(X)^T W^{(0)} + b^{(0)}) \tag{1}$$

$$Z^E = Relu(\widetilde{Z}^E D^{-1/2}(A(1 - \beta) + I\beta)D^{-1/2} W^{(1)} + b^{(1)}) \tag{2}$$

where $W^{(0)}$ is the weight matrix in the first layer and $W^{(1)}$ is the weight matrix in the second layer.

The adjacency matrix and feature matrix’s dot product is used to compute the aggregate of neighbouring nodes’ features during convolution. An identity matrix, or “ I ” is added to the adjacency matrix in order to include a node’s self-features. The terms $I\beta$ and $A(1 - \beta)$ indicate the different weights given to the node’s self-features and neighbours, respectively. The letter “ D ” stands for the diagonal matrix of A . The adjacency matrix is calculated in the preprocessing phase as $(A(1 - \beta) + I\beta)$, where β denotes the weights given to the features of the node and its neighbours.

4.2.2. Structure Reconstruction Decoder

The node embeddings produced by the encoder are fed into the decoder for structure reconstruction, which then reconstructs the original network structure.

$$A_{rec} = sigmoid(Z^E(Z^E)^T) \tag{3}$$

It is possible to calculate the likelihood that an edge will appear between any two possible node pairs by computing the inner product between embeddings. Subsequently, the reconstruction error is calculated using the following formula:

$$A_{loss} = A - A_{rec} \tag{4}$$

4.2.3. Attribute Reconstruction Decoder

To reconstruct node attributes, an additional convolutional layer is utilized to predict the input node attribute in the following manner:

$$X_{rec} = Relu(Z^E D^{-1/2}(A(1 - \beta) + I\beta)D^{-1/2} W^{(2)} + b^{(2)}) \tag{5}$$

The reconstruction error is computed using the following procedure:

$$X_{loss} = X - X_{rec} \tag{6}$$

4.2.4. Loss Function

The objective of NodeAnomEn algorithm is to minimize the following function:

$$L_{loss} = (1 - \alpha)A_{loss} + \alpha X_{loss} \quad (7)$$

Here, the term A_{loss} represents the error in reconstructing the network structure, while X_{loss} denotes the error in reconstructing the node attributes.

4.2.5. Anomaly Detection

The node's anomalous score is determined by adding the structure reconstruction error and the attribute reconstruction error.

$$anomaly_score(v_i) = (1 - \alpha)\|a - \hat{a}_i\|_2 + \alpha\|X_i - \hat{X}_i\|_2 \quad (8)$$

The anomaly score for each node is determined by a balance between the attribute error and adjacency matrix reconstruction error, controlled by the parameter α . Nodes with higher anomaly scores are identified as anomalies, and every node is ranked according to their computed anomaly score. This approach is influenced by techniques that aim to detect anomalies in a node feature subspace, as discussed in [3].

4.3. Edge Anomaly Detection Task

Edge anomaly detection is based on the following principle. The opposite of outlier edge identification is the detection of missing edges which is also known as link prediction [22]. The aim is to find the missing edges between pair of nodes in the graph.

The goal of the algorithm is to calculate the anomaly score for each edge in the graph. The score is based on the similarity between nodes. Edges with low similarity between the two end nodes are more likely to be outliers. The algorithm calculates the anomaly score for each edge by measuring the similarity of the nodes (as illustrated in Algorithm 2). After the model is trained, it predicts the label (positive or negative) for each possible edge in the graph. A positive label indicates the presence of an edge between two nodes, while a negative label signifies the absence of an edge. Ideally, the graph should only contain positive edges. If any negative edges are present, they are considered anomalies and indicate deviations from the normal patterns in the network.

This section introduces the proposed Edge Anomaly Encoder (EdgeAnomEn) for edge anomaly detection using the proposed methodology. The proposed architecture is shown in Figure 2. EdgeAnomEn employs an unsupervised learning mechanism based on auto encoder. It consists of: (i) Two layers of AnomEn in encoder. (ii) One dot product decoder.

Algorithm 2: Edge anomaly detection procedure

Input : Adjacency matrix A, Feature matrix X, existing edge labels L_e

Output: Set of anomalous edges A

$h_v^0 \leftarrow X_v, \forall v \in \mathcal{N}(v);$

$\tilde{Z}^E = \text{Relu}\left(D^{-1/2}(A(1 - \beta) + I\beta)D^{-1/2}(X)^T W^{(0)} + b^{(0)}\right);$

$Z^E = \text{Relu}\left(\tilde{Z}^E D^{-1/2}(A(1 - \beta) + I\beta)D^{-1/2}W^{(1)} + b^{(1)}\right);$

$edge_predict = Z^E(Z^E)^T;$

$edge_label = \text{generate_labels}()$

foreach Edge **do**

if $edge_predict == \text{'Negative'}$ and $edge_label \neq \text{'Negative'}$ **then**

 | Issue alert for anomalous edge;

end

end

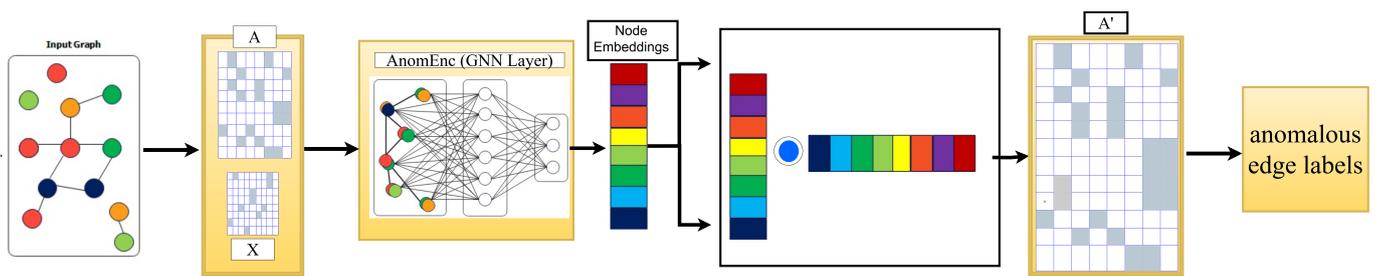


Figure 2. The Proposed Framework (EdgeAnomEn) for Edge Anomaly Detection in Attributed Networks.

Network Encoder

A total of two convolutional layers are used to map the input to node embeddings, Z^E , which can be formulated as follows:

$$\tilde{Z}^E = Relu(D^{-1/2}(A(1 - \beta) + I\beta)D^{-1/2}(X)^T W^{(0)} + b^{(0)}) \tag{9}$$

$$Z^E = Relu(\tilde{Z}^E D^{-1/2}(A(1 - \beta) + I\beta)D^{-1/2} W^{(1)} + b^{(1)}) \tag{10}$$

where $W^{(0)}$ is the weight matrix in the first layer and $W^{(1)}$ is the weight matrix in the second layer.

In this equation, D is the diagonal degree matrix of the graph, A is the adjacency matrix, I is the identity matrix, X is the input feature matrix, β is a hyper parameter that balances the importance of the adjacency matrix and the identity matrix, $Relu$ is the rectified linear unit activation function, $W^{(0)}$ is the weight matrix of the first convolutional layer, and $b^{(0)}$ is the bias term of the first convolutional layer.

The encoder creates the node embeddings using the network structure and attribute information as input. The decoder then uses the input of these node embeddings to reconstruct the network topology. Decoder predicts the label for all possible edges as either positive edge or negative edge. Positive edge means the edge between the respective nodes are present in the graph. Negative edges means that the edge between the respective two end nodes is very unlikely.

Subsequently, the predicted edge label is compared with the existing edge label. If the predicted label and existing label bear a difference in an edge, then that edge is considered as anomalous.

5. Experimental Setup

This section describes the experiments performed on real world attributed networks to empirically evaluate the proposed method. The proposed graph representation learning methodology is examined on both node and edge anomaly detection tasks.

5.1. Experimental Setup for Node Anomaly Detection Task

The method has been proven to yield performance enhancements in both scenarios, and was evaluated on the Twitter [23], Enron, and Amazon datasets during the study. The number of nodes, attributes, and anomalous nodes used for each dataset can be found in Table 2. The twitter dataset includes 1950 human accounts that have been verified and 3000 fake accounts, which were procured from websites that specialize in selling such accounts. However, only 100 fake accounts were ultimately included in the dataset, which represents roughly 5% of the authentic user base. Additionally, the dataset contains anomalous nodes equivalent to 5% of the regular nodes. The architecture employed in this study features two convolutional layers, with the first layer having a size of 256. Table 3 presents the AUC values for various embedding dimensions of the second layer. The Enron and Amazon datasets have a learning rate of 0.0005, while the learning rate for the Twitter dataset is 0.0133 multiplied by 0.001. The β value used for all three datasets is 0.8.

Table 2. AUC scores of NodeAnomEn (proposed) vs. DOMINANT [18].

Layer Size	Twitter		Enron		Amazon	
	DMNT	RAD	DMNT	RAD	DMNT	RAD
256	0.5737	0.5757	0.710	0.7661	0.626	0.634
128	0.6074	0.6119	0.7310	0.7310	0.626	0.643
64	0.630	0.6401	0.7310	0.7722	0.626	0.6343
32	0.6283	0.6308	0.7310	0.7680	0.626	0.6344
16	0.6376	0.6394	0.7310	0.7680	0.626	0.6344
8	0.5987	0.6000	0.7310	0.7310	0.626	0.6349

Table 3. Base method vs. EdgeAnomEn comparison: accuracy and AUC for edge anomaly detection on PolitiFact and GossipCop datasets.

Model	Politifact		Gossipcop	
	Accuracy	AUC	Accuracy	AUC
GNN CL	0.6018	0.7257	0.9339	0.9509
GCN FN	0.8597	0.9020	0.9638	0.9636
UPFD (GCN-Encoder)	0.8235	0.8842	0.9045	0.9153
UPFD (GAT-Encoder)	0.8100	0.8937	0.9241	0.9343
UPFD (SAGE)	0.8462	0.8859	0.9723	0.9722
EdgeAnomEn (proposed)	0.9129	0.9388	0.9798	0.9796

The following methods are used for comparison:

LOF [24]: “Identifying Density-Based Local Outliers” presents a novel approach for detecting local outliers in datasets using density-based clustering techniques. The authors propose a metric called the Local Outlier Factor (LOF), which measures the degree of outlierness of a data point with respect to its local neighborhood. The LOF metric is based on the concept of the local density of a data point relative to the densities of its neighbors.

DOMINANT [18]: The paper “Deep Anomaly Detection on Attributed Networks” proposes a method for detecting anomalous behavior in attributed networks using deep learning techniques. The authors use a two-stage approach that first learns a node embedding using a Graph Convolutional Network (GCN) and then uses an Autoencoder to detect anomalies in the learned embedding.

Scan [25]: A structural clustering algorithm for networks by Xu et al. (2007) presents a novel algorithm for clustering nodes in complex networks based on their structural properties. The algorithm, called SCAN (Structural Clustering Algorithm for Networks), is designed to identify clusters that exhibit high internal connectivity and low external connectivity, known as “communities” in the network analysis literature.

AMEN [26]: Scalable Anomaly Ranking of Attributed Neighborhoods by Perozzi and Akoglu (2016) proposed a new method for detecting anomalies in attributed networks. The method is based on the observation that anomalies often manifest themselves as neighborhoods in the network that have unusual patterns of attributes (such as node features or edge weights). The authors’ approach involves first selecting a set of “anchor nodes” that are representative of different parts of the network. For each anchor node, they then construct an attributed neighborhood consisting of the node itself and its neighbors within a certain distance. They use a dimensionality reduction technique (specifically, Singular Value Decomposition or SVD) to transform the high-dimensional attribute vectors of the nodes in each neighborhood into a lower-dimensional space.

5.2. Experimental Setup for Edge Anomaly Detection Task

The proposed EdgeAnomEn architecture, which competes with the existing approaches, makes use of the proposed encoder Anom-En that was introduced in part II. Here, first layer and second layer have used layer size of 128 and 64, respectively.

GNN-CL [23]: This method presents a graph neural network (GNN) with continual learning for detecting fake news from social media. The proposed framework leverages both node and edge information in the social network and achieves state-of-the-art performance on two publicly available datasets. The approach is robust to concept drift, and the authors suggest that it has the potential to be applied to other tasks in social network analysis.

The proposed architecture has produced the best results when compared to the current approaches as shown in Table 3. The proposed method is compared with the existing methods UPFD-GCN [17], GCN-FN [27], GNN-CL [23], UPFD-GAT [28], UPFD-SAGE [29]. GNN-CL is a GNN developed specifically for graph classification which encodes the news propagation graph. Profile information and textual embeddings of comments are considered user features.

6. Performance Evaluation & Results

Kaize Ding et al. [18] reported the AUC scores for various anomaly detection methods, including LOF [24], SCAN [25], AMEN [26], Radar [30], Anomalous [31], and Dominant [18], on the ‘BlogCatalog’ dataset. The AUC scores for these methods were 0.4915, 0.2727, 0.6648, 0.7104, 0.7281, and 0.7813, respectively. It is evident that the Dominant method outperformed all other methods, and thus, it was selected as the reference method for comparison. Table 3 illustrates the AUC scores for the proposed method and reference methods on the Twitter, Enron, and Amazon datasets.

The performance of the proposed method and the baseline method on the Twitter dataset was compared in Figure 2 for various embedding vector sizes (8, 16, 32, 64, 128, and 256). The results showed that the proposed method achieved the best AUC value of 0.6401, while the baseline method achieved an AUC value of 0.630 when the dimensions of the first and second layers were set to 256 and 64, respectively. The proposed method consistently outperformed the baseline method across all other layer dimensions, as indicated by Figure 2.

Figure 3 presents the comparison of the proposed method and the baseline method on the Enron dataset for the same embedding vector sizes. The best AUC value of 0.7722 was obtained for NodeAnomEn, while the AUC value for DOMINANT was 0.7310, when the dimensions of the first and second layers were set to 256 and 64, respectively. The proposed method showed superior performance to the baseline method across the Enron dataset, as illustrated by Figure 3. Based on the AUC values obtained from the experiments, the proposed method outperforms the baseline methods on the Enron dataset by approximately 5.48%.

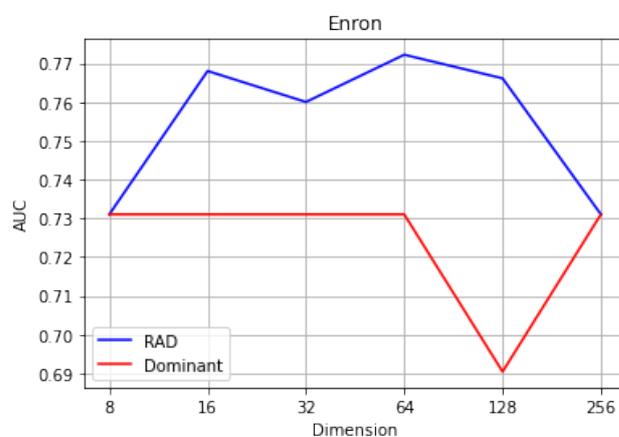


Figure 3. NodeAnomEn vs. DOMINANT.

Similarly, Figure 4 compares the proposed method and the baseline method on the Amazon dataset for embedding vector sizes of 8, 16, 32, 64, 128, and 256. The results

demonstrated that the proposed method outperformed the baseline method, achieving the best AUC value of 0.6344 compared to the AUC value of 0.6260 for DOMINANT, when the first and second layer dimensions were set to 256 and 32, respectively. Figure 5 shows that the proposed method consistently outperforms the baseline method across all other layer dimensions.

Therefore, the proposed method outperforms the baseline method on all three datasets for the node anomaly detection task.

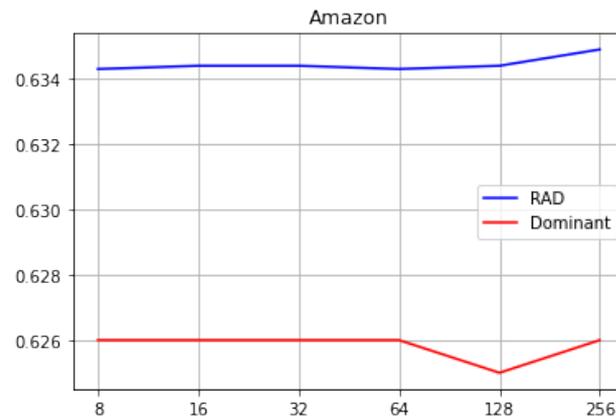


Figure 4. NodeAnomEn vs. DOMINANT.

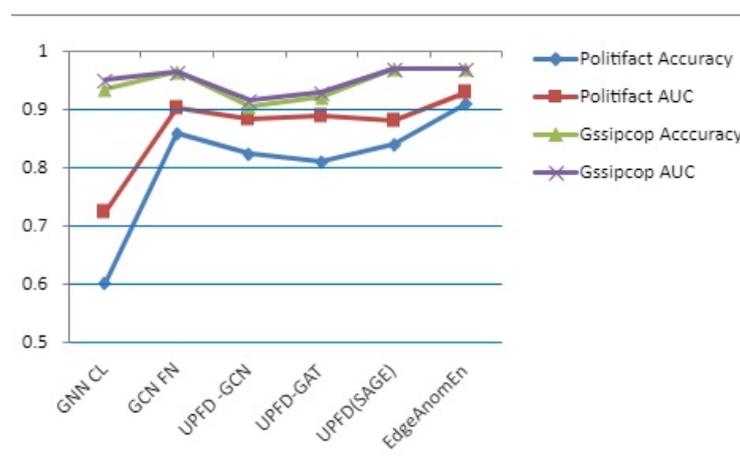


Figure 5. Comparison of base methods and proposed method EdgeAnomEn in terms of accuracy and AUC for edge anomaly detection on PolitiFact and GossipCop datasets. The figure displays the performance trend of the different methods, with higher values indicating better performance.

Table 3 presents the performance evaluation results of various models, specifically on two datasets—PolitiFact and Gossipcop. The evaluation metrics used in this study are accuracy and area under the curve (AUC).

For the PolitiFact dataset, the results show that the proposed model, EdgeAnomEn, outperforms all other models, achieving the highest accuracy score of 0.9129 and AUC score of 0.9388. The next best performing model is UPFD(SAGE), achieving an accuracy score of 0.8462 and AUC score of 0.8859.

For the Gossipcop dataset, the proposed model, EdgeAnomEn, again performs the best, achieving the highest accuracy score of 0.9798 and AUC score of 0.9796. The second-best performing model is UPFD(SAGE), achieving an accuracy score of 0.9723 and AUC score of 0.9722.

The results suggest that the proposed model, EdgeAnomEn, is highly effective for both datasets and outperforms the other models in all evaluation metrics. UPFD(SAGE) also performs well on both datasets, but falls short of EdgeAnomEn in terms of accuracy and AUC scores.

It is evident that EdgeAnomEn exhibits superior performance as compared to all the baseline models.

7. Conclusions

The task of identifying anomalous users on attributed social networks is a challenging problem that requires the integration of network structure and node attributes. Graph Neural Networks (GNNs) are well-suited for this task. However, the use of aggregate operations in GNNs can lead to false positives and false negatives. The AnomEn approach, a robust graph neural network developed for anomaly detection, addresses these challenges using a weighted aggregate mechanism that places greater emphasis on a node's own features and less emphasis on its neighbors' features. The proposed AnomEn method serves as the encoder in node and edge anomaly detection architectures and outperforms existing methods in both tasks. The findings demonstrate the effectiveness and robustness of the proposed method in detecting anomalies on attributed networks.

Author Contributions: Methodology, G.V.D.; Investigation, P.P.; Supervision, K.C. and V.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Department of Science and Technology-Interdisciplinary Cyber Physical Systems (DST-ICPS), Government of India, under DST/ICPS/Cluster/Data Science/2018/General, Project ID: T-739.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Noble, C.C.; Cook, D.J. Graph-based anomaly detection. In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 24–27 August 2003; pp. 631–636.
2. Bronstein, M.M.; Bruna, J.; LeCun, Y.; Szlam, A.; Vandergheynst, P. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Process. Mag.* **2017**, *34*, 18–42. [[CrossRef](#)]
3. Hassanzadeh, R.; Nayak, R.; Stebila, D. Analyzing the effectiveness of graph metrics for anomaly detection in online social networks. In Proceedings of the International Conference on Web Information Systems Engineering, Paphos, Cyprus, 28–30 November 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 624–630.
4. Hassanzadeh, R.; Nayak, R. A rule-based hybrid method for anomaly detection in online-social-network graphs. In Proceedings of the 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, Herndon, VA, USA, 4–6 November 2013; pp. 351–357.
5. Hassanzadeh, R.; Nayak, R. A semi-supervised graph-based algorithm for detecting outliers in online-social-networks. In Proceedings of the 28th Annual ACM Symposium on Applied Computing, Coimbra, Portugal, 18–22 March 2013; pp. 577–582.
6. Sun, J.; Qu, H.; Chakrabarti, D.; Faloutsos, C. Neighborhood formation and anomaly detection in bipartite graphs. In Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05), Houston, TX, USA, 27–30 November 2005; p. 8.
7. Akoglu, L.; McGlohon, M.; Faloutsos, C. Oddball: Spotting anomalies in weighted graphs. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Hyderabad, India, 21–24 June 2021; Springer: Berlin/Heidelberg, Germany, 2010; pp. 410–421.
8. Wang, A.H. Do not follow me: Spam detection in twitter. In Proceedings of the 2010 IEEE International Conference on Security and Cryptography (SECRYPT), Athens, Greece, 26–28 June 2010; pp. 1–10.
9. Sun, J.; Xie, Y.; Zhang, H.; Faloutsos, C. Less is more: Compact matrix decomposition for large sparse graphs. In Proceedings of the 2007 SIAM International Conference on Data Mining, Minneapolis, MN, USA, 26–28 April 2007; SIAM: Philadelphia, PA, USA, 2007; pp. 366–377.
10. Miller, B.A.; Beard, M.S.; Bliss, N.T. Eigenspace analysis for threat detection in social networks. In Proceedings of the 14th IEEE International Conference on Information Fusion, Chicago, IL, USA, 5–8 July 2011; pp. 1–7.
11. Yang, W.; Shen, G.W.; Wang, W.; Gong, L.Y.; Yu, M.; Dong, G.Z. Anomaly detection in microblogging via co-clustering. *J. Comput. Sci. Technol.* **2015**, *30*, 1097–1108. [[CrossRef](#)]
12. Liu, Y.; Chawla, S. Social media anomaly detection: Challenges and solutions. In Proceedings of the Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, Australia, 10–13 August 2015; pp. 2317–2318.
13. Sun, H.; Huang, J.; Han, J.; Deng, H.; Zhao, P.; Feng, B. gskeletonclu: Density-based network clustering via structure-connected tree division or agglomeration. In Proceedings of the 2010 IEEE International Conference on Data Mining, Sydney, Australia, 13–17 December 2010; pp. 481–490.
14. Henaff, M.; Bruna, J.; LeCun, Y. Deep convolutional networks on graph-structured data. *arXiv* **2015**, arXiv:1506.05163.

15. Niepert, M.; Ahmed, M.; Kutzkov, K. Learning convolutional neural networks for graphs. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 20–22 June 2016; pp. 2014–2023.
16. Zhang, S.; Tong, H.; Xu, J.; Maciejewski, R. Graph convolutional networks: A comprehensive review. *Comput. Soc. Netw.* **2019**, *6*, 1–23. [[CrossRef](#)]
17. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.
18. Ding, K.; Li, J.; Bhanushali, R.; Liu, H. Deep anomaly detection on attributed networks. In Proceedings of the 2019 SIAM International Conference on Data Mining, Calgary, AB, Canada, 2–4 May 2019; SIAM: Philadelphia, PA, USA, 2019; pp. 594–602.
19. Zhang, G.; Li, Z.; Huang, J.; Wu, J.; Zhou, C.; Yang, J.; Gao, J. Efraudcom: An e-commerce fraud detection system via competitive graph neural networks. *Acm Trans. Inf. Syst. (TOIS)* **2022**, *40*, 1–29.
20. Daniel, G.V.; Venkatesan, M. Robust graph based deep anomaly detection on attributed networks. In Proceedings of the 2021 11th IEEE International Conference on Cloud Computing, Data Science & Engineering (Confluence), Uttar Pradesh, India, 28–29 January 2021; pp. 1029–1033.
21. Li, J.; Dani, H.; Hu, X.; Liu, H. Radar: Residual Analysis for Anomaly Detection in Attributed Networks. In Proceedings of the IJCAI, Melbourne, Australia, 19–25 August 2017; pp. 2152–2158.
22. Zhang, H.; Kiranyaz, S.; Gabbouj, M. Outlier edge detection using random graph generation models and applications. *J. Big Data* **2017**, *4*, 1–25. [[CrossRef](#)]
23. Han, Y.; Karunasekera, S.; Leckie, C. Graph neural networks with continual learning for fake news detection from social media. *arXiv* **2020**, arXiv:2007.03316.
24. Breunig, M.M.; Kriegel, H.P.; Ng, R.T.; Sander, J. LOF: Identifying density-based local outliers. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, TX, USA, 16–18 May 2000; pp. 93–104.
25. Xu, X.; Yuruk, N.; Feng, Z.; Schweiger, T.A. Scan: A structural clustering algorithm for networks. In Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, CA, USA, 12 August 2007; pp. 824–833.
26. Perozzi, B.; Akoglu, L. Scalable anomaly ranking of attributed neighborhoods. In Proceedings of the 2016 SIAM International Conference on Data Mining, Miami, FL, USA, 5–7 May 2016; SIAM: Philadelphia, PA, USA, 2016; pp. 207–215.
27. Monti, F.; Frasca, F.; Eynard, D.; Mannion, D.; Bronstein, M.M. Fake News Detection on Social Media using Geometric Deep Learning. *arXiv* **2019**, arXiv:1902.06673.
28. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
29. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. *arXiv* **2017**, arXiv:1706.02216.
30. Peng, Z.; Luo, M.; Li, J.; Liu, H.; Zheng, Q. ANOMALOUS: A Joint Modeling Approach for Anomaly Detection on Attributed Networks. In Proceedings of the IJCAI, Stockholm, Sweden, 13–19 July 2018; pp. 3513–3519.
31. Sánchez, P.I.; Müller, E.; Irmeler, O.; Böhm, K. Local context selection for outlier ranking in graphs with multiple numeric node attributes. In Proceedings of the 26th International Conference on Scientific and Statistical Database Management, Aalborg, Denmark, 30 June–2 July 2014; pp. 1–12.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.