Node And Edge Anomaly Detection in Social Networks Using SVM-Clustering and Graph Neural Network Models

[1] Yallamanda Rajesh Babu, [2] Dr. G. Karthick, [3] Dr. V.V. Jaya Rama Krishnaiah

[1]Research Scholar, Dept. of Computer Science and Engineering
Annamalai University
Annamalainagar – 608002
Tamilnadu, India
[2]Assistant Professor, Dept. of Computer Science and Engineering
Annamalai University
Annamalainagar – 608 002
Tamilnadu, India
[3]Associate Professor, Dept. of Computer Science and Engineering
ASN Women's Engineering College Tenali,
Andhra Pradesh.

Email: [1]yrajeshbabu7@gmail.com, [2]karthick18588@gmail.com, [3]jkvemula@gmail.com

Abstract: Anomaly detection is an essential subject with numerous applications and has thus been investigated for decades in numerous research fields, including cyber security, finance, healthcare, social networks, etc. In recent years, numerous strategies have been utilized to detect anomalies in unstructured or multidimensional data. Graphs have the expressive capacity to model data from several domains, including social networks. Literature has a significant amount of research on finding anomalies utilizing structural attributes, ego nets for subspace selection/community analysis. These learning techniques are called shallow mechanisms because they do not completely understand graph structures or patterns. This technique cannot capture the intricate interaction between network nodes and other information modalities. Therefore, the capability of deep learning to solve the challenge of detecting anomalies in social networks was utilized. This article provides a methodology for detecting anomalies in graph data, such as social networks, using graph neural networks' robust representation capability. This paper proposed a model for identifying anomalous connections between social network nodes and edges using deep learning techniques. Clustering and traditional One-class SVM graph neural networks for node anomaly identification was used, and tuned the graph neural networks for edge anomaly detection. The proposed framework produces superior outcomes in comparison to previous baseline models.

Keywords - Social Networks, Anomaly Detection, Deep Learning, Graph Representation Learning, Graph Neural Networks.

1. Introduction

The network is a powerful tool for locating objects and relationships and connecting them meaningfully to portray real-time entities as objects. There are many examples of interconnected systems in the real world: road networks that link cities, financial networks that link banks worldwide, and social networks that link users, businesses, and customers through relationships such as friendship. This can be conveniently modeled as graphs, with the real-time entities serving as the graph's nodes. These kinds of networks are sometimes referred to as attributed networks because, in addition to the network structure, the objects in real-time also have attributes or characteristics. The expressive capability of graphs allows them to be used to model and analyze data from many other domains; for instance, in a social network like Facebook or Twitter, where the nodes are simply users and the edges are the relationships between them.

In such networks, anomaly detection, the process of identifying objects or relationships that differ from the remainder, is a significant challenge. Sharing, communication, and collaboration are facilitated by social media platforms. Numerous malicious activities, such as cyber bullying, the planning of terrorist attacks, and the dissemination of fraudulent information, are highly likely. Consequently, it is essential to detect these anomalous actions as quickly and precisely as feasible in order to reduce their occurrence.

Many statistical and machine learning [4][14][26] approaches have been proposed to tackle this issue, but they are ineffective because these rely on superficial learning techniques that are incapable of understanding the interconnections and relationships found in graph organized (non-euclidean) data. Several methods exist in traditional machine learning for spotting anomalies, such as Principal Component Analysis (PCA) [17] and one-class SVM [15]. For this outlier detection task, SVM's of a certain type have seen extensive use. It is necessary to generalize these traditional methods to this graph structured data to perform anomaly detection jobs. One of the most powerful machine learning methods to separate typical data from outliers is Support Vector Data Description (SVDD) [23]. However, these techniques are only used to Euclidean data in the literature. Many connections between nodes and node/edge attributes play an important role in the learning phase of graph structured data. Traditional graph learning algorithms like DeepWalk. [20] Often overlook feature information, which focus instead on acquiring the graph's structural information.

Graph embedding has recently proven to be an effective strategy to learn low-dimensional network representations that capture and retain graph structure. As a result, there is a lot of focus on graph representation learning, with methods from several types of neural networks being applied. The standard method involves training an anomaly detection model with a feature vector of fixed length for each node, where each feature represents the structural information of the node's immediate neighbor in the graph.

Both DeepWalk [20] and node2vec [9] are static graph embedding methods that rely on random walks. These methods are effective at resolving the neighbourhood structure and providing a more complete image of a node. In contrast, this ignores feature information, which is a major flaw of such superficial learning approaches. Recent advances in graph representation learning include Graph Neural Networks (GNNs) [25], which are able to overcome the drawbacks of shallow learning approaches while still yielding many superior results. These methods, which borrow heavily from the concept of message forwarding, can be applied straight to graph information. In order to build a new embedding, each node takes the vectors it receives from its neighbours and uses a variety of techniques to aggregate them. After several iterations, each node has a feature vector that details the local environment in terms of structure and attributes. These methods, such as GCN [13] and GAT [24], are becoming increasingly popular in various tasks, such as node classification, thus it is essential to take advantage of their potential to address GAD issues. This paper discusses graph neural network techniques, which have proven effective at simultaneously representing graph structure and attributes in recent years.

In this research, the powerful representation of GNNs, the idea of one-class SVM and clustering to identify anomalous nodes was used.

2. Literature Survey

This section provides a quick overview of the research conducted in this field. First, a complete description of traditional graph anomaly detection methods is provided, followed by a discussion of graph representation learning and deep learning techniques for anomaly detection.

2.1 Anomaly detection with traditional statistical and machine learning techniques

In conventional anomaly detection with non-euclidean data, such as graphs, the only available information is the data's structure and, in some cases, the node/edge properties. Therefore, anomaly detection systems must rely on the network's structural information to identify patterns that may be anomalous. Without explicit characteristics, these algorithms must use the graph representation to derive a usable collection of features connected to the graph structure. Several characteristics related to nodes, ego nets [1][2] and global graph structure can be constructed and utilized for anomaly detection activities. Node-level characteristics may include, for instance, node degrees, closeness, betweenness centralities, number of common neighbors, etc.

One of the most well-known structure-based methods is ODDBALL, which was introduced by [1]. To quickly find ego nets that don't follow the same patterns (called Anomalous ego nets), this method uses the extracted ego net attributes to try to uncover patterns that most ego nets adhere to. Many community-based methods, like [3][4] Local Outlier Factor and [14] Isolation Forest, relied on the local point density to generate their predictions.

One can find regions with a similar density and places with a significantly lower density than their neighbors, referred to as outliers. AUTOPART is one such method based on the notion that nodes with different representations/neighborhoods will be clustered separately from nodes with the same representation/neighborhood. In [26] SCAN, vertices that share numerous neighbors are sorted into the same clusters; vertices that serve as a bridge between many clusters are referred to as hubs, and vertices that do not belong to any of the communities are identified as outliers.

[19] Divide this anomaly identification problem into two subproblems: identifying abnormal substructures in graph data and identifying abnormal subgraphs among a specified set of subgraphs whose nodes and edges have characteristics. Occasionally, some characteristics are more significant or contribute to making a node unusual. [16][17][18] Developed a system (GOUTRANK) for outlier detection that ranks every node. EdgeCentric: [21][22]. In this study, rather than relying excessively on node attributes or graph structural information, the authors utilize edge attributes/information to identify edge-attributed anomalies in graph data. Specifically, this method uses Minimum Description Length (MDL) to score abnormalities of nodes based on unsupervised patterns of edge-attribute activity.

2.2 Graph Neural Networks

GNNs generalize the application of deep learning to graph data. To graph-structured data, GNNs have been derived from the concept of convolutional neural networks. It employs a message-passing architecture in which messages are sent between graph nodes, and neural networks are used for updating. Each node compiles data from its neighbors and develops a new feature vector. This procedure is repeated k times for a node to obtain structural information from its k-hop neighbors. Therefore, these latent representations of nodes, learned by GNNs, are employed for tasks such as node classification, link prediction, etc. [13] The Graph Convolutional Network (GCN) model introduced by [12] is the state-of-the-art GNN model. GCN produced better outcomes than conventional shallow graph representation learning algorithms like DeepWalk. [24] Graph Attention Network (GAT) introduced GCN's masked self-attention layers. This alters how GCN aggregates data from its neighbors. GAT specifies a mechanism to give local neighbors weight by assigning them various weights without bias. [10][11] GraphSAGE is an additional inductive GNN-based technique that works well with new, unseen points. It can forecast the embeddings of new nodes without the need for retraining. All current GNN approaches, however, are focused on learning embedded representations of nodes, and it is unclear how to utilize the potential of GNNs for GAD.

2.3 Anomaly Detection with Deep Learning Techniques

Deep learning and graph representation learning techniques have been used to solve anomaly detection problems in recent years. These methods substantially outperform the prior benchmarks established by the Local outlier factor. Four LOF or twenty-six SCAN algorithms. DOMINANT combines these unsupervised Graph convolutional networks with an autoencoder to find anomalies in attributed networks. The inaccuracy in graph reconstruction is minimized, and an anomaly score is computed for each node. [8] Anomaly DAE A concept comparable to DOMINANT that employs a dual auto-encoder-based system for anomaly identification. Also, at the encoding phase, a self-attention mechanism has been implemented so that various weights will be assigned to neighborsto effectively capture the structural patterns of the network. [5][6][7] As a semi-supervised learning technique, Castellini built a de-noising Autoencoder for anomaly identification. This semi-supervised learning with an autoencoder does not require overtly aberrant training samples. StreamSpot detects anomalies in a stream of heterogeneous graphs with diverse types of nodes and edges. They have developed a new similarity function for comparing two heterogeneous graphs based on the relative frequency of their local substructures and a centroid-based clustering technique for capturing the typical Behaviors. Again, MIDAS focuses on the stream of edges and assigns them anomaly ratings in real-time and with constant memory. In network traffic data, they mostly detect micro-clusterabnormalities, sudden clusters of suspiciously similar edges, such as lockstep behavior and denial of service attacks. [27] NETWALK is a further proposed Auto Encoder based approach for dynamic networks that encodes graph data by producing random network walks. These nodes/edges are then grouped to identify data anomalies.

3. Proposed Methodology

Complex systems are modelled extensively using graph-structured data in multiple domains. While there are ways to identify anomalous instances in Euclidean data, discovering anomalies in graph structured data is a possible issue. This section focuses on developing a model to detect anomalous nodes and edges in social networks. To service this anomaly detection goal, the power of approaches was utilized in graph representation learning, such as GCN. First, it was provided a complete description of node anomaly detection, followed by the task of edge anomaly detection.

a. In this entire work, let G be an attributed graph such as G= (V , E) where V = v_1, v_2, ..., v_N is the set of N = |V | nodes and E ⊆V X V is the set of edges (M = |E |) with the associated binary adjacency matrix, mainly named as A∈R|n×n| where A_{ij} = 1 if there is an edge between vertex v_i and v_j else 0.
b. It also assumed that the techniques require (not mandatory) a real-valued matrix of node features/attributes X ∈R|n×f| (e.g. Bag of words representation of the text features in case of citation networks or the individual attributes in case of social networks like Facebook). The aim is to use the information contained in A and X to map each node to a vector z ∈R|f|, where f « | V |. Table 1 show the notations used in node and edge anomaly detection.

3.1 Problem and Notations

Given the anomaly-free training dataset x_i, i= 1, ..., K, the model is trained to cluster the normal data together, and then the model produces the anomaly score S_{xu} for an unseen data point x_u. A data point with a high anomaly score is defined as ananomaly. All nodes for training belong to one class (normal), while the remaining validation and testing nodes are unlabelled. On the other hand, given the trainingdataset with positive edges and the same number of anomalous edges, the model is trained to learn these embeddings to differentiate between anomalous and non-anomalous data and verify the results during the testing and validation datasets, including unseen edges and anomalous edges.

Table 1 Notation for Graph-based Anomaly Detection

Notations	Description
$V=v_1,v_2,..,v_N$	These to f N nodes in a graph
$V_{tr} \subseteq V$, $\mid V_{tr}\mid =K$	The set of K training nodes
$X \in R^{N\times F}$	Node feature matrix
$A \in R^{N\times N}$	Adjacency matrix
$Z \in R^{N\times F}$	Node embedding matrix
$g(H^{(l)},AW^{(l)})$	GCN propagation rule at layer
θ_k	Slack variable
$\underline{R}$	Radius to determine the anomaly

3.2 Anomalous Node Detection

The overall framework of node anomaly detection has been divided into two phases: The graph embedding phase and then the clustering phase.

Graph Embedding Phase

The adjacency and the feature matrix are fed into this step. The strength of graph neural networks was used to incorporate the graph. It was tested three cutting-edge graph neural networks, including GCN, GAT, and GraphSAGE. These methods take the input as graph depicted in Figure 1 and output embeddings for each node. Now it is seen that normal data is all required during training for a one-class SVM. After obtaining all of the nodes' embeddings, it is then hided the original embeddings during training.

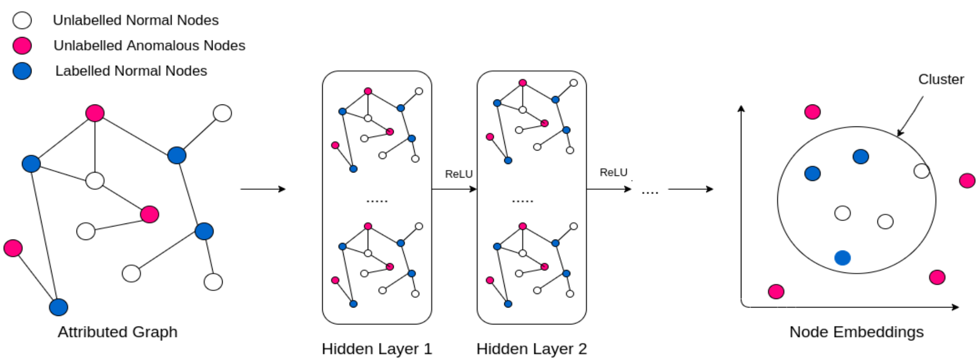

Figure 1 Overall framework of the proposed methodology

Clustering Phase

The normal data embeddings from phase 1 are now available. Now, the attempt is to group these points into a cohesive whole by bringing them as close as possible to the cluster's epicentre. Since, the data get partitioned during training; it has only typical data points to work with. To do this, the model was given with both typical and out-of-the-ordinary data points during the evaluation phase, and then the attempt was made to squeeze all the points into the cluster. After training is complete, this model should be able to tell the difference between typical and out-of-the-ordinary node embeddings.

3.2.1 Learning Objective

The application of node anomaly detection uses only typical nodes as training data points. Therefore, all those data points can be clustered together, and this model can determine a boundary. The original goal of SVDD learning was to decrease the hypersphere's radius by as much as possible around the normal points.

$$min_r \left(r^2 + \frac{1}{\beta K}\sum_{i=1}^{k} \xi i\right) \qquad (1)$$

Such that

$$\|\theta_k(x_i) - c \quad \|^2 F_k \leq r^2 + \xi i \xi j \geq 0, \forall i$$

Where, ξi is a slack variable to allow some of the boundary points to be termed as outliers in the training dataset to avoid overfitting the data. The data points θk(xi) Fk are not all strictly inside the hypersphere, but the data located too far away must be penalised which is the purpose of this slack variable $\beta \in$ 0,1. After minimizing the equation (1), center c and the radius r can be obtained. Data points that lie outside the hypersphere ,$\|\theta_k(x_i) - c \quad \|^2 F_k \leq r^2$, are outliers.

Now this was the learning objective of the classic SVDD. We need to incorporate the graph neural network while finding out the embeddings and the clustering part to find the radius. GCN takes the input matrix A and the node features X to get the embeddings. This has layer wise trainable weights in GNN W_1, W_2. . . W_l where l belongs to the number of hidden layers. For lth layer, the forward propagation rule is defined by the equation (2).

$$H(l+1) = g(H(l), A; W (l)) \qquad (2)$$

The aim of the model is to jointly learn the network parameters W, minimise the distance of the data points inside the cluster characterised by the radius r and the centroid c. Given a graph defined by (X, A) and the set of K training nodes,

$$L(r, W) = \frac{1}{\beta K}\sum_{v_i \in v_{tr}}\left[\|g(X, A: W)v_i - c\|^2 - r^2\right]^{+} + r^2 + {}^{\lambda}\!/_{2}\sum_{t=2}^{L}\|W^{(l)}\|^{2} \qquad (3)$$

The first term in equation (3) is the penalty for nodes lying on/outside the radius, if the distance between an embedding vector and the center c is greater than the radius r. As in classical SVDD, the second term, minimizing r^2, is to minimize the volume of sphere. The last term is the weight decay regularizer on the network parameters W with a hyperparameter λ > 0. The objective lets the network learn to map the node embeddings that are closed to the center c of the sphere. Since the training nodes are all normal, the model will extract the common factors of the given nodes. As a result, the description boundary of normal nodes can be obtained and the anomalous nodes can be detected. For a node v_i in the given graph, its anomaly score S (v_i) can be defined by the location of the embedding respect to the sphere: S (v_i) = $||$g(X, A; W)v_i −

c|| ^{2}r^2 . If S $(v_i) > 0$, the node v$_i$ is anomalous, otherwise it is a normal node. In the validation and the test dataset, this has anomalous and non - anomalous nodes. So the model performance can be evaluated on these validation and test datasets. Like other standard neural networks, this use SGD to optimize the parameters of GNN.

The initial center of the cluster is determined by the K- means clustering algorithm, where the initial embeddings are given to the algorithm to find out the centroid. Since, radius r is not an inner parameter of the network, r and W can not be optimized synchronously by the BP algorithm. Instead, we update the r and W alternately during the training phase. First, we train W for $\theta \in N$ epochs while r is fixed. After every θ the epochs, r can be solved by simple linear percentile search. That is, for the training node set V_{tr} , we can obtain a distance set $dV_{tr} \in R^K$. Afterward, we can sort d_{Vtr} from small to large, and the radius r can be defined via $(1 - \beta)$ percentile of d_{Vtr}

3.3 Anomalous Edge Detection

Graph Neural Networks (GNN's) like Graph Convolutional Networks (GCN's) are demonstrated on the task of node classification in citation networks. While in this work, these GNN are tuned to solve the edge detection problem by modeling them as a supervised learning algorithm. This task of edge anomaly detection is primarily divided into 3 stages:

1. Graph Embedding Algorithms
2. Edge Embedding Algorithms
3. Link Classification Layer

As it is shown that, the best results come from graph neural networks like GCNs and GATs regarding graph embedding. In this work, the same method has been applied to learning graph structured data. The source destinations pairs of an edge was passed to GNNs to obtain the source and destination embeddings. Once the embeddings of the nodes in the graph is obtained, it is needed to generate the edge embeddings in order to pass it to the next stage. To generate the edge embeddings from source and destination nodes, two standard techniques was used that have been found in the literature: 1. Hadamard Product. 2. Concatenation of source and destination nodes. Once the edge embeddings were obtained, these edges were passed embeddings to the dense classification layer, which will obtain the probabilities of the possible edge existence between it. Experimentally, concatenation of source and destination vertices found to be slightly better than the hadamard product. The overall architecture is seen in Figure 2.

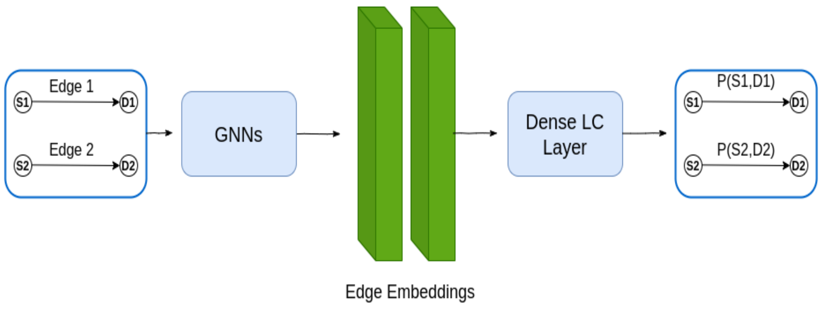

Figure 2 Edge Anomaly Detection Architecture

Each and every part of the model was trained by minimizing the loss function. The ADAM algorithm adjusts the model parameters based on a training set of links, allowing for binary cross entropy to be calculated between the predicted link probabilities and the genuine link labels.

4. Implementation Details And Results

This section discusses the specifics of the experiments that have been performed on the suggested models. This model is divided into two parts: experiments aimed at detecting anomalies at nodes and experiments aimed at detecting anomalies at edges.

4.1 Node Anomaly Detection

To evaluate the model’s performance, the standard citation networks like CORA, and PubMed II was taken. Citation networks are nothing but papers citing other papers in the network. Each scientific publication is represented as a node in the graph, and the citations between two publications are the edges / relationship between the nodes. These networks also have associated node features, which are the bag of word representations computed from the dictionary of total words.

Table 2 Datasets for Node Anomaly Detection

Datasets	Nodes	Edges	Features	Train/Val/Test
Cora	2708	5429	1433	490/246/410
PubMed II	19717	44338	500	4725/2364/3936

The node anomaly detection datasets used in this work are generated from three plain node classification datasets by regarding one class as normal and the rest as anomalous. The normal classes in Cora and Pubmed datasets are “Neural Networks” and “Diabetes Mellitus Type 2”, respectively. All the nodes in the training set pertain to the normal class, while the validation and testing set has both normal and anomalous classes as well (50-50% each).

Table 3 Performance of Node Anomaly Detection Models

	Model	**CORA(AUC)**	**PUBMED(AUC)**
Raw Features	IForest	53.09	65.57
	OCSVM	54.05	45.5
	PCA	62.17	71.06
	AE	62.17	71.05
DeepWalk	IForest	57.87	60.73
	OCSVM	52.1	60.22
	PCA	55.9	61.66
	AE	55.91	61.66
GAE Based	GCN-AE	80.53	58.26
	GAE	60.15	54.27
	DOM	67.5	53.93
GNN NA	GCN NA	84.29	66.25
	GAT NA	**93.17**	64.54
	SAGE NA	87.01	**76.12**

All three of the most cutting edge anomaly detection techniques in machine learning DeepWalk, principal component analysis, and isolation forest have been compared. It was also compared the model's results to those obtained using deep autoencoder based approaches, such as DOMINANT, simple GAE, and GCN-AE. Table 3 presents results for node anomaly identification on the CORA and PUBMED datasets using the suggested model.

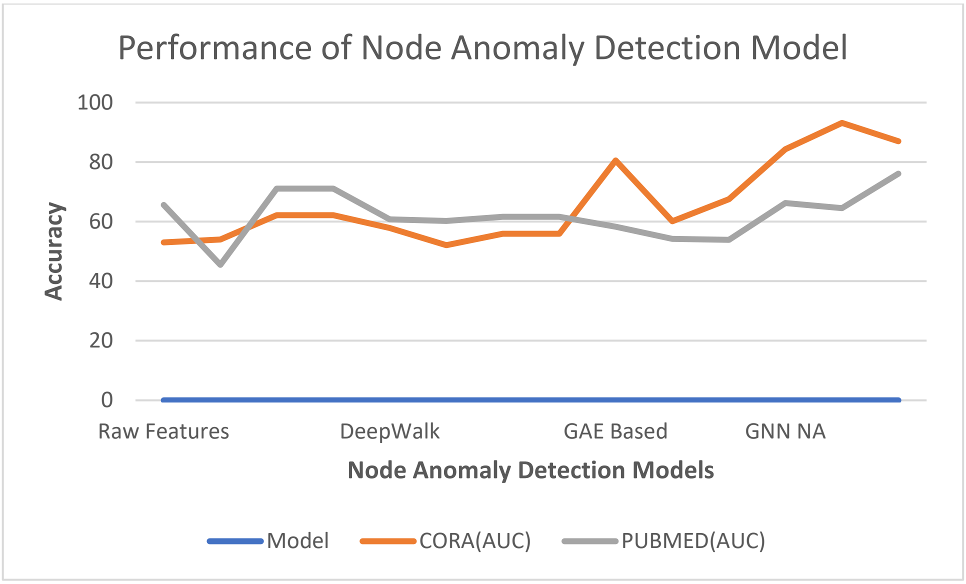

Figure 3 Performance Comparison of Proposed Model with State of the art Models

In this framework, three popular GNNs have been used to test the performance of the model, GCN [13,] GAT [24] and GraphSAGE [11] and the configurations for these three models are almost the same. Figure 3 compares the proposed model's AUC performance to that of several state-of-the-art node anomaly detection models. The slack variable beta has been set to 0.1 in all the experiments. The initial weight matrices of the GNNs are initialized using glorot uniform weight initialization. ADAM optimizer is used with learning rate as 0.001. This applies two layers GNN for cora dataset with hidden layer size as 16-16 and for pubmed dataset 32-32. ReLU is used as an activation function in the GNN models with 0.5 dropout. Attention heads in GAT is set as 8. Aggregation method in GraphSAGE is pooling and LSTM aggregators for CORA and PubMed datasets respectively. Any other modification may lead to even better performance and still can be improved.

4.2 Edge Anomaly Detection

The models are trained on an incomplete version of the datasets where part of the citation links have been removed, while all the node features are kept as it is. Validation and test sets are formed from previously removed edges and the same number of randomly sampled pairs of unconnected nodes (Anomalous edges). Model comparison has been done basedon their ability to correctly classify non-anomalous edges and anomalous edges.

Table 4 Datasets for Edge Anomaly Detection

Datasets	**Nodes**	**Edges**	**Features**
Cora	2708	5429	1433
Citeseer	3327	4732	3703

Table 5 Performance of Edge Anomaly Detection Models

	CORA(AUC)	**Citeseer(AUC)**
Spectral Clustering (SC)	84.6	80.5
DeepWalk	83.01	80.5
VGAE	91.4	90.8
GNN Edge Anomaly	GCN EA 86.57	GCN EA 79.18
	GAT EA **93.20**	GAT EA **91.19**

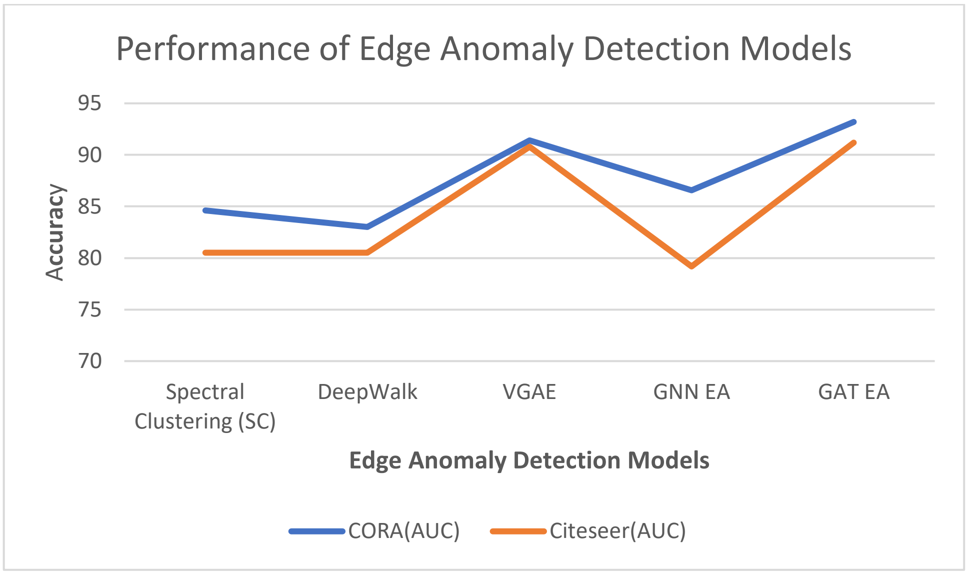

Figure 4 Performance Comparison of Proposed Edge Anomaly Detection Model with State of the art Models

Since, the framework is based on supervised learning, initially anomalous edges are randomly created in the training phase (equal to the number of normal edges). These edges, with the edges originally present in the graph are fed to the model in the training phase. This model has been tested on some standard link prediction work like spectral clustering, Deepwalk and variational graph autoencoder. Table 5 presents results for edge anomaly identification on the CORA and citeseer datasets using the suggested model. Figure 3 compares the proposed model's AUC performance to that of several state-of-the-art edge anomaly detection models.

5. Conclusion

In this work, a model is proposed for node and edge anomaly detection in social networks using deep learning approaches. This also did a comprehensive analysis of the anomalies exist in the graph data like social networks and the work that has been done in this area of study. The node anomaly detection approach was based on classic One-class SVM graph neural networks and clustering, while the edge anomaly detection was achieved by tuning the graph neural networks for anomaly detection. The results from the extensive experiments demonstrate that proposed models achieves significant improvements. This work is done assuming the data in hand is a static graph data. In the future, it can be generalized to finer graph level and dynamic node / edge anomaly detection task.

References

[1] Akoglu, L., Mcglohon, M., and Faloutsos, C. Oddball: Spot- ting anomalies in weighted graphs. pp. 410–421.
[2] Akoglu, L., Tong, H., and Koutra, D. Graph-based anomaly detection and description: A survey, 2014.
[3] Bhatia, S., Hooi, B., Yoon, M., Shin, K., and Faloutsos, C. Midas: Micro cluster-based detector of anomalies in edge streams. In AAAI Conference on Artificial Intelligence (AAAI) (2020).
[4] Breunig, M., Kriegel, H.-P., Ng, R. T., and Sander, J. Lof: Identifying density-based local outliers. In proceedings of the 2000 ACM sigmod international conference on management of data (2000), ACM, pp. 93–104.
[5] Castellini, J., Poggioni, V., and Sorbi, G. Fake twitter followers detection by denoising autoencoder. pp. 195–202.
[6] Chakrabarti, D. Autopart: Parameter-free graph partitioning and outlier detection. In Knowledge Discovery in Databases: PKDD 2004 (Berlin, Heidelberg, 2004), J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, Eds., Springer Berlin Heidelberg, pp. 112–124.
[7] Ding, K., Li, J., Bhanushali, R., and Liu, H. Deep Anomaly Detection on Attributed Networks. pp. 594–

602.

[8] Fan, H., Zhang, F., and Li, Z. Anomalydae: Dual autoencoder for anomaly detection on attributed networks, 2020.

[9] Grover, A., and Leskovec, J. Node2vec: Scalable feature learning for networks, 2016.

[10] Hamilton, W. L. Graph representation learning. Synthesis Lectures on Artificial Intelligence and Machine Learning 14, 3, 1–159.

[11] Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs, 2018.

[12] Kipf, T. N., and Welling, M. Variational graph auto-encoders, 2016.

[13] Kipf, T. N., and Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the 5th International Conference on Learning Representations (2017), ICLR ’17.

[14] Liu, F. T., Ting, K. M., and Zhou, Z.-H. Isolation forest. In 2008 Eighth IEEE International Conference on Data Mining (2008), pp. 413– 422.

[15] Manevitz, L., and Yousef, M. One-class SVMs for document classification. Journal of Machine Learning Research 2 (01 2001), 139– 154.

[16] Manzoor, E. A., Momeni, S., Venkatakrishnan, V. N., and Akoglu, L. Fast memory-efficient anomaly detection in streaming heterogeneous graphs, 2016.

[17] Mac´ Kiewicz, A., and Ratajczak, W. Principal Components Analysis (pca). Computers Geosciences 19, 3 (1993), 303–342.

[18] Mu¨ Ller, E., Sa´ Nchez, P. I., Mu¨ Lle, Y., and Bo¨ Hm, K. Ranking outlier nodes in subspaces of attributed graphs. In 2013 IEEE 29th International Conference on Data Engineering Workshops (ICDEW) (2013), pp. 216–222.

[19] Noble, C., and Cook, D. Graph-based anomaly detection. pp. 631– 636.

[20] Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk. Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (Aug 2014).

[21] Savage, D., Zhang, X., Yu, X., Chou, P., and Wang, Q. Anomaly detection in online social networks, 2016.

[22] Shah, N., Beutel, A., Hooi, B., Akoglu, L., Gunnemann, S., Makhija, D., Kumar, M., and Faloutsos, C. Edge centric: Anomaly detection in edge-attributed networks, 2015.

[23] Tax, D., and Duin, R. Support vector data description. Machine Learning 54 (01 2004), 45–66.

[24] Velicˇ Kovic´, P., Cucurull, G., Casanova, A., Romero, A., Lio` , P., and Bengio, Y. Graph attention networks. In International Conference on Learning Representations (2018).

[25] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems 32, 1 (Jan 2021), 4–24.

[26] Xu, X., Yuruk, N., Feng, Z., and Schweiger, T. A. J. Scan: a structural clustering algorithm for networks. In KDD ’07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (New York, NY, USA, 2007), ACM, pp. 824– 833.

[27] Yu, W., Cheng, W., Aggarwal, C. C., 0001, K. Z., Chen, H., and 0010, W. W. NetWalk: A Flexible Deep Embedding Approach for Anomaly Detection in Dynamic Networks. In Proceedings of the 24th International Conference on Knowledge Discovery and Data Mining (2018), ACM, pp. 2672–2681.