IT414 Project Report on

Outlier detection using Unsupervised Graph Neural Network

Submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

by

Aprameya Dash (191IT209)

Pratham Nayak (191IT241)

Sarthak Jain (191IT145)

under the guidance of

Dr. Nagamma Patil

DEPARTMENT OF INFORMATION TECHNOLOGY

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

SURATHKAL, MANGALORE - 575025

April 2023

DECLARATION

We hereby *declare* that the IT414 Project Report entitled ***"Outlier detection using Unsupervised Graph Neural Network"*** , which is being submitted to the **National Institute of Technology Karnataka, Surathkal**, for the award of the Degree of Bachelor of Technology in Information Technology, is a *bonafide report of the work carried out by us.* The material contained in this IT414 Project Report has not been submitted to any University or Institution for the award of any degree.

(1) Aprameya Dash (191IT209)

(2) Pratham Nayak (191IT241)

(3) Sarthak Jain (191IT145)

Department of Information Technology
Place : NITK, Surathkal
Date : 03/04/2023

CERTIFICATE

This is to *certify* that the IT414 Project Report entitled ***"Outlier detection using Unsupervised Graph Neural Network"*** submitted by

(1) Aprameya Dash (191IT209)

(2) Pratham Nayak (191IT241)

(3) Sarthak Jain (191IT145)

as the record of the work carried out by them, is *accepted as the B.Tech. IT414 report submission* in partial fulfillment of the requirement for the award of degree of Bachelor of Technology in Information Technology in the Department of Information Technology, NITK Surathkal.

Dr. Nagamma Patil
Assistant Professor
Department of Information Technology
NITK Surathkal

ABSTRACT

Outlier detection plays an important part in various statistical technologies; thus, accurate outlier detection is necessary for statistics and mathematics. Outliers are defined as noisy data, and often ignoring the behaviour of such noisy data can cause statistical models to make wrong decisions in various situations. Thus, there is an urgent need to devise methodologies to identify and label such data points accurately. Although there are different outlier detection algorithms, such as clustering algorithms, these algorithms often fail to detect outliers located nearby dense clusters or hidden among normal data points or objects. Thus, to identify such difficult-to-find data objects, this paper proposes a novel methodology that employs a graph neural network model based on GraphSAGE and trained in an unsupervised method. The proposed methodology creates a graph based on the similarity of data objects with their neighbourhood and then uses weighted information from the neighbourhood in order to learn newer embeddings of the data objects. These new embeddings learned via unsupervised training make it easier to highlight and identify outliers that are generally unidentifiable using conventional means. The proposed methodology is evaluated using eight different datasets from the *Outlier Detection DataSets (ODDS)* and results reveal that the proposed methodology outperforms the existing methodologies in seven of the eight datasets and sets a new milestone in the field of outlier detection.

Keywords— GNN, Graph Neural Network, GraphSAGE, ODDS, Outlier Detection

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Overview

Outliers are defined as data objects that behave abnormally as compared to a majority of the remaining data objects. These objects often add unnecessary noise and inconsistency to traditional statistical techniques resulting in erroneous outcomes, and thus, there exists a necessity to identify and discard such abnormal data objects in most statistical algorithms. Outlier detection algorithms are special algorithms that are designed to locate such abnormal data objects. Most of the commonly used conventional outlier detection algorithms consider the data objects independently and use measures such as the density of data objects, the distance between data objects etc., in order to label data objects as outliers. But these algorithms usually do not consider the relationships of the data objects with their own neighbourhood to identify outliers and thus fail to properly identify outliers which are close to dense clusters of data objects or which are hidden among normal data objects. To incorporate the neighbourhood information while performing outlier detection, this paper uses graph neural networks.

Graph neural networks (GNNs) are neural networks that are specifically meant to process and extract topological information from the graph structure of the given input. Regular neural networks, such as dense layers, are trained to only extract features from the given data point. However, graph neural networks extract information from not only the given data point but also from the neighbourhood of the data point when represented as a vertex in a graph. They can be directly applied to various types of graphs and provide an easy way to do node-level, edge-level, and graph-level predictions. Some of the standard GNNs that are widely used are GCN (Kipf and Welling, 2016), GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2017), etc. GraphSAGE is a framework for inductive representation learning on large graphs. SAGE stands for sampling and aggregation, which are the two major steps involved. GraphSAGE is used to generate low-dimensional vector representations for nodes and is especially useful for graphs that have rich node attribute information.

However, the standard GraphSAGE convolution operation does not support weighted edges. Hence we use a custom GNN layer based on GraphSAGE convolution along with added support for edge weights to perform outlier detection. Since in real-life applications, most of the data is unlabelled, outlier detection techniques should be unsupervised. Hence this project develops an unsupervised learning-based technique to perform outlier detection using a modified version of GraphSAGE.

1.2 Organization of the report

The report consists of 5 chapters. The first chapter gives a brief introduction to the problem and the proposed approach. The second chapter describes in detail the existing works on outlier detection and also formulates the problem statement and objectives. The third chapter describes the working of the proposed approach in detail. The fourth chapter tabulates the results obtained on various ODDS datasets along with detailed analysis and comparison with existing works. The fifth chapter consists of the conclusion and future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Background and related works

Many research works have focused on solving the task of outlier detection over the years. Most of the earlier works used statistical means in order to identify outliers. Many other research works also focus on using density-based and distance-based methods to identify outliers. Further, more recent works have focused on using machine learning-based models for outlier detection. Also a combination of different outlier detection algorithms is also being used to create more robust models.

Statistical techniques generally make an assumption regarding the underlying distribution model of the entire data and then use different supervised, unsupervised and semi-supervised ways to identify the data objects which violate the assumed underlying distribution model. Tang et al. (2015) combines Gaussian mixture model (GMM) and Locality preserving projections (LPP) techniques to create a better separation between the normal data objects and the outliers by preserving the structure of the neighbourhood of data objects. Saha et al. (2009) proposes a method based on principal component analysis (PCA) in order to perform outlier detection and automate active contours for the task of object detection. The proposed method performs better than the other outlier detection algorithms. Park and Jeon (2015) create a regression-based model in order to detect outliers. The work considers the observed values to synthesise an independent variable or signal via a weighted summation method and create a regression model using this signal, thus, achieving great results in an artificial dataset.

Density-based and distance-based methods are some of the oldest methods used for outlier detection. The underlying declaration is that normal data objects should be present in dense neighbourhoods, whereas outliers should be present in low-density areas. Breunig et al. (2000) proposes a method which assigns a value called local outlier factor (LOF) to each object which gives a degree to which that object can be an outlier and depends on the isolation and the neighbourhood of the data object. Vázquez et al. (2018) proposes a new density-based algorithm called SDO (Sparse

Data Observers) to measure the degree to which a data object is an outlier. The algorithm greatly decreases the computational resources required for performing outlier detection. Zhang et al. (2009) proposes a new method which assigns a Local Distance-based Outlier Factor (LDOF) to each object to measure its outlier-ness by considering the location-based relationship of an object with its neighbours.

Clustering-based methods are also popularly used for outlier detection. In these methods, generally, smaller clusters containing very few data objects are declared as abnormal data objects. Clustering algorithms like K-Means (MacQueen (1967)), DBSCAN (Ester et al. (1996)), HPStream (Aggarwal et al. (2004)), DCluster (Zhang et al. (2005)), CHAMELEON (Karypis et al. (1999)) etc are commonly employed for this purpose. Many of the recent works focus on the creation of machine learning models to identify outliers, as these models are generally more robust and perform better than other techniques. This has also led to the creation of ensembled outlier detection models. Zhao and Hryniewicki (2019) proposes DCSO, which is an unsupervised outlier detector combination framework, to dynamically identify and combine the top-performing outlier detector models. Thus, this framework gives better results than other static combination approaches.

Recently, graph neural networks have presented a new way of incorporating neighbourhood information and analysing data in the form of graphs. Du et al. (2022) proposes a novel outlier detection methodology based on graph neural network structure. The usage of graph structure allows this research work to effectively incorporate the neighbourhood information of data objects while identifying outliers. Evaluation of eight real-world datasets reveals that the proposed method made great leaps in outlier detection. Hamilton et al. (2017) introduced GraphSAGE, which is a graph neural network model meant for large graphs. GraphSAGE uses sampling to create small subgraphs that are used to aggregate neighbourhood information of the nodes. Hence the name SAGE, which stands for sampling and aggregation. Different permutation invariant functions can be used to perform aggregation, such as mean aggregation, summation-based aggregation or any other permutation invariant function. Thus, such graph neural network-based approaches can be used to perform outlier detection as outliers can be detected by analysing the neighbourhoods of data objects properly. Further, there are very few works which focus on using graph neural network-based approaches for tackling outlier detection, thus, presenting ample chances for future research.

2.2 Problem statement

To identify the outliers for a given set of data points using techniques based on graph neural networks and unsupervised learning.

2.3 Objectives of the project

(1) Effectively representing the given data points using a graph structure.

(2) Building a graph neural network architecture that captures neighbourhood information to compute embeddings for each of the data points/nodes.

(3) Formulating an unsupervised loss function to train the graph neural network architecture.

(4) Formulating a relation to compare the embeddings with the data points to identify the outliers.

CHAPTER 3

PROPOSED METHODOLOGY

3.1 Graph construction

3.1.1 Graph representation

The given set of data points is represented using a graph structure by interpreting the points as the vertices of a graph. For each pair of vertices/data points, an edge is added between them. These edges are weighted directed edges with an edge weight that is computed using the cosine similarity between the vector representation of the two data points. Eq. (3.2) shows the equation to compute the cosine similarity between two vectors.

$$similarity_{cosine}(a, b) = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}} \tag{3.1}$$

3.1.2 Graph Sampling

Sampling of a graph refers to selecting a suitable subgraph that will be used to train the model in a memory-efficient way. One way of sampling is by randomly sampling a fixed number of neighbours for each node. Randomized sampling makes the model more robust and generalized. Another way of sampling is by sampling the neighbours of a node based on some metric/measure. Since we have a weighted graph, the weights of the edges are used to sample the neighbourhood of any given node. Let K be a constant that defines the number of sampled neighbours. Then for each node, the K adjacent nodes with the highest cosine similarity are sampled. Hence after sampling, the complete graph is transformed into a graph where each node has an in-degree of K. Since the nodes with the highest cosine similarity are picked to represent the neighbourhood, it decreases the chances of having an edge between a normal data point and an outlier, thereby making the graph representation more effective.

3.2 GNN architecture

3.2.1 Batch normalization layer

Batch normalization refers to z-score normalization. Batch normalization allows us to use much higher learning rates, which further increases the speed at which networks train. Let V be the set of data points, x_v be the initial vector representation of node v, then the output after batch normalization is given by Eq. (3.2), where $E[V]$ is the mean of the inputs samples, $Var[V]$ is the variance of the input samples, β and γ are learnable parameters.

$$h_v^{(1)} = \frac{x_v - E[V]}{\sqrt{Var[V]}} * \gamma + \beta \tag{3.2}$$

3.2.2 Input dense layer

The input is then passed through a feed-forward layer to extract relevant features from the normalized input vectors as shown in Eq. (3.3), where $W^{(2)}$ and $b^{(2)}$ are learnable parameters and σ is a non-linear activation function.

$$h_v^{(2)} = \sigma(h_v^{(1)} \cdot W^{(2)} + b^{(2)}) \tag{3.3}$$

3.2.3 Graph convolution layers

Let the neighbourhood of node v be given by $N(v)$. The graph convolution operator used for neighbourhood aggregation in the l^{th} GraphSAGE layer when summation-based aggregation is used is given by Eq. (3.4), where $W_{self}^{(l)}, W_{neigh}^{(l)}$ and $b^{(l)}$ are learnable parameters and σ is a non-linear activation function.

$$h_v^{(l)} = \sigma(W_{self}^{(l)} \cdot h_v^{(l-1)} + W_{neigh}^{(l)} \cdot \sum_{u \in N(v)} h_u^{(l-1)} + b^{(l)}) \tag{3.4}$$

Let e_{uv} be the weigh of the edge between node u and v, then using the modified equations, the embedding of the node v in the l^{th} layer is given by Eq. (3.5).

$$h_v^{(l)} = \sigma(W_{self}^{(l)} \cdot h_v^{(l-1)} + W_{neigh}^{(l)} \cdot \sum_{u \in N(v)} e_{uv} \cdot h_u^{(l-1)} + b^{(l)}) \tag{3.5}$$

Two layers of the modified GraphSAGE convolution operators are used to extract neighbourhood information from the nodes as shown in Eq. (3.6) and (3.7), where $W_{self}^{(3)}, W_{neigh}^{(3)}, b^{(3)}, W_{self}^{(4)}, W_{neigh}^{(4)}$ and $b^{(4)}$ are learnable parameters and σ is a non-linear activation function.

$$h_v^{(3)} = \sigma(W_{self}^{(3)} \cdot h_v^{(2)} + W_{neigh}^{(3)} \cdot \sum_{u \in N(v)} e_{uv} \cdot h_u^{(2)} + b^{(3)}) \tag{3.6}$$

$$h_v^{(4)} = \sigma(W_{self}^{(4)} \cdot h_v^{(3)} + W_{neigh}^{(4)} \cdot \sum_{u \in N(v)} e_{uv} \cdot h_u^{(3)} + b^{(4)}) \tag{3.7}$$

3.2.4 Output dense layer

Finally, the mean of the output of the two graph convolutional layers is passed through another feed-forward layer to get the node's final embeddings which will then be used to detect outliers. This is shown in Eq. (3.8), where $W^{(5)}$ and $b^{(5)}$ are learnable parameters and σ is a non-linear activation function.

$$h_v = \sigma(\frac{(h_v^{(3)} + h_v^{(4)})}{2} \cdot W^{(5)} + b^{(5)}) \tag{3.8}$$

3.3 Unsupervised learning

The parameters of the proposed GNN model are trained using an auto encoder-inspired, unsupervised loss function given by Eq. (3.9). The loss function essentially tries to minimize the Euclidean distance between the embeddings and the initial data points. Since the number of outliers is small, the model will learn a mapping function that maps the majority of the points (non-outliers) to an embedding that is close to the original data point, whereas the outliers will be mapped further away.

$$Loss(x_v, h_v) = \sqrt{\sum_{i=1}^{n} |x_i - h_i|^2} \tag{3.9}$$

3.4 Outlier detection

For a given set of points, first, the graph is constructed and sampled based on cosine similarity scores between the nodes. Then the proposed model is trained on the constructed graph using the unsupervised loss function. The model is then used to generate the embeddings for each of the nodes of the graph (data points). The Euclidean distance between the data points and the computed embeddings is calculated, and the top n data points with the highest euclidean distance with their embeddings are reported as the outliers.

CHAPTER 4

RESULT AND ANALYSIS

4.1 Dataset description

The proposed method for performing outlier detection is trained and evaluated on the Outlier Detection DataSets (Rayana (2016), Dua and Graff (2017)). ODDS is a large collection of outlier detection datasets and also includes the ground truths of the datasets, if available. To enable comparison with existing works, the proposed outlier detection method is trained and evaluated on eight datasets selected from ODDS. The selected datasets, along with their details, are listed in Table 4.1.1. Each dataset contains a list of multidimensional coordinate vectors, each representing a data object and the ground truths for each of the data objects, given by the label 0 or 1, with outliers labelled as 1.

Table 4.1.1: Dataset description

Sl No.	Name of Dataset	No. of samples	No. of features	No. of outliers	Outlier Ratio
1	WBC	377	30	20	5.30 %
2	Wine	129	13	10	1.20 %
3	Pima	768	8	268	34.80 %
4	Glass	214	9	9	4.20 %
5	Vertebral	240	6	13	12.50 %
6	Ionosphere	351	33	127	36.10 %
7	Vowels	1452	12	46	3.10 %
8	Letter	1600	32	100	6.25 %

4.2 Experimental setup

The unsupervised training and evaluation for the proposed method are carried out on the Google Colab platform using an Intel(R) Xeon(R) CPU operating at 2.20GHz with around 12GB of RAM. The proposed model is trained in an unsupervised manner on each of the results, and then evaluation is carried out on each dataset. For each dataset, the hyper-parameters, K and learning rate are varied, and the best-performing set of values are noted down. Table 4.2.1 shows the optimum hyper-parameters selected for each dataset.

Table 4.2.1: Hyperparameters used in the different datasets

Sl No.	Name of Dataset	Learning Rate	K
1	WBC	0.005	10
2	Wine	0.01	10
3	Pima	0.005	20
4	Glass	0.01	20
5	Vertebral	0.01	10
6	Ionosphere	0.01	10
7	Vowels	0.01	40
8	Letter	0.01	40

4.3 Evaluation Metric

To evaluate the performance of the proposed model in outlier detection, the AUC Score metric is considered. AUC Score is defined as the area under the receiver operating characteristic (ROC) curve. To calculate the AUC score, first, the ROC curve needs to be constructed. For this, a quantity, *contamination level*, is defined, which gives the fraction of data points which are outliers. Next, the ROC curve is drawn by plotting the False Positive Rate (FPR) on the x-axis and True Positive Rate (TPR) on the y-axis by varying the contamination level from 0 to 1. At any value of contamination level, the True Positive Rate (TPR) represents the fraction of positive samples that are correctly classified, and False Positive Rate (FPR) is the fraction of negative samples that are incorrectly classified. Equations 4.1 and 4.2 display the formulae for calculating TPR and FPR, respectively, when the number of True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN) are given.

$$TPR = \frac{TP}{TP + FN} \tag{4.1}$$

$$FPR = \frac{FP}{FP + TN} \tag{4.2}$$

4.4 Experimental Results

The proposed graph neural network-based architecture is trained and evaluated on each of the eight datasets selected from the ODDS collection. Next, the ROC curve is constructed by evaluating the model for different contamination levels for each dataset, and the AUC score is computed and recorded. Table 4.4.1 shows the comparison of the obtained AUC score with the existing state-of-the-art techniques. The comparison reveals that the proposed system performs better on seven of the eight selected datasets. Thus the proposed approach can be termed as the new state of the art.

Table 4.4.1: Performance on ODDS datasets

Sl No.	Name of Dataset	Proposed Model AUC Score	State of the art AUC Score
1	WBC	**0.98**	**0.98**
2	Wine	**1.00**	**1.00**
3	Pima	0.77	**1.00**
4	Glass	**0.96**	0.93
5	Vertebral	**0.84**	0.67
6	Ionosphere	**0.96**	0.88
7	Vowels	**0.95**	0.93
8	Letter	**0.91**	0.89

Table 4.4.2 shows the comparison with the proposed approach when no neighbourhood information is used. This is achieved by setting K = 0. As evident from the results, removing neighbourhood information reduces the performance drastically, thereby emphasizing the importance of graph structure.

Table 4.4.2: Performance on ODDS datasets without graph structure

Sl No.	Name of Dataset	Proposed Model AUC Score (K $\neq$ 0)	Proposed Model AUC Score (K = 0)
1	WBC	**0.98**	0.97
2	Wine	**1.00**	**1.00**
3	Pima	**0.77**	0.72
4	Glass	**0.96**	0.88
5	Vertebral	**0.84**	0.70
6	Ionosphere	**0.96**	0.95
7	Vowels	**0.95**	0.93
8	Letter	**0.91**	0.85

CHAPTER 5
CONCLUSIONS AND FUTURE WORK

Outlier detection is an integral part of various domains such as data analysis, statistics, computational science, data processing etc. As such, there is a need for outlier detection systems which can accurately identify outliers from a given set of data objects. Most of the existing works in this field do not leverage the use of neighbourhood information of data objects to identify outliers and, thus, perform poorly when the outliers are situated near dense clusters or in normal object regions. In this paper, an unsupervised GraphSAGE-based technique is proposed to improve the accuracy of outlier detection. The dataset consisting of multiple data points is converted into a graph representation, and edges are created between neighbours using a cosine similarity-based mechanism. Next, the proposed GraphSAGE-based model is trained on the constructed graph using an unsupervised auto-encoder-inspired loss function to generate embeddings for nodes (data points) of the graph. The inclusion of neighbourhood information and auto-encoder-inspired unsupervised learning enables the model to generate embeddings for data points in such a way that it becomes easier to separate out the outliers. Evaluation of the proposed model on eight datasets from the ODDS collection reveals that the proposed model outperforms the existing works by getting better AUC scores in seven of the eight datasets. The research work can be further improved by devising a methodology to automate the value of K (number of most similar neighbours from which information flows to a given data point) as it is currently a hyper-parameter which needs to be set manually and experimented with to get the most optimum results. Further, instead of GraphSAGE, other graph neural architectures can be used for experimentation.

REFERENCES

Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu. A framework for projected clustering of high dimensional data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 852–863, 2004.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

Xusheng Du, Jiong Yu, Zheng Chu, Lina Jin, and Jiaying Chen. Graph autoencoder-based unsupervised outlier detection. *Information Sciences*, 608:532–550, 2022.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *computer*, 32(8):68–75, 1999.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

J MacQueen. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pages 281–297. University of California Los Angeles LA USA, 1967.

Chang Mok Park and Jesung Jeon. Regression-based outlier detection of sensor measurements using independent variable synthesis. In *Data Science: Second International Conference, ICDS 2015, Sydney, Australia, August 8-9, 2015, Proceedings 2*, pages 78–86. Springer, 2015.

Shebuti Rayana. Odds library, 2016.

Baidya Nath Saha, Nilanjan Ray, and Hong Zhang. Snake validation: A pca-based outlier detection method. *IEEE signal processing letters*, 16(6):549–552, 2009.

Xiu-ming Tang, Rong-xiang Yuan, and Jun Chen. Outlier detection in energy disaggregation using subspace learning and gaussian mixture model. *International Journal of Control and Automation*, 8(8):161–170, 2015.

Félix Iglesias Vázquez, Tanja Zseby, and Arthur Zimek. Outlier detection based on low density models. In *2018 IEEE international conference on data mining workshops (ICDMW)*, pages 970–979. IEEE, 2018.

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

Ji Zhang, Wynne Hsu, and Mong Li Lee. Clustering in dynamic spatial databases. *Journal of intelligent information systems*, 24:5–27, 2005.

Ke Zhang, Marcus Hutter, and Huidong Jin. A new local distance-based outlier detection approach for scattered real-world data. In *Advances in Knowledge Discovery and Data Mining: 13th Pacific-Asia Conference, PAKDD 2009 Bangkok, Thailand, April 27-30, 2009 Proceedings 13*, pages 813–822. Springer, 2009.

Yue Zhao and Maciej K Hryniewicki. Dcso: dynamic combination of detector scores for outlier ensembles. *arXiv preprint arXiv:1911.10418*, 2019.