Extracting Spatiotemporal Features For Detecting the Beginning of a Network Layer Attack with a Graph Neural Autoencoder and Deep Metric Learning

Mukesh Yadav
Tandy School of Computer Science
College of Engineering and Computer Science
The University of Tulsa
Tulsa, Oklahoma, USA
muy8752@utulsa.edu

Peter Hawrylak
Tandy School of Computer Science
College of Engineering and Computer Science
The University of Tulsa
Tulsa, Oklahoma, USA
peter-hawrylak@utulsa.edu

2024 Cyber Awareness and Research Symposium (CARS) | 979-8-3503-8641-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/CARS61786.2024.10778706

***Abstract*—Detection and mitigation of ongoing network layer attacks from real-time network traffic data is essential for normal operations and availability of the interconnected devices in a network. Our study focuses on detecting the presence of network layer attacks from simultaneously available network flow data using a model that utilizes Graph Neural Networks (GNN), Long Short-Term Memory (LSTM), deep metric learning, and Support Vector Machine (SVM). The GNN-LSTM model extracts spatiotemporal features from network traffic segments. A deep metric learning method is utilized to enhance the differentiation between normal and attack traffic data, which are then classified using SVM. The proposed approach outperforms the existing machine learning models on a publicly available dataset, which shows the effectiveness of the GNN-LSTM model in detecting network layer attacks.**

***Index Terms*—Graph Neural Network, Anomaly Detection, Graph Dataset, Deep Metric Learning.**

I. Introduction

Network layer attack possesses challenges to provide services to the users. Various techniques are used for detecting network layer attacks. Signature based detection, rule-based detection, Graph Neural Network (GNN), deep learning, and ensemble methods are techniques used for detecting network layer attacks. Detection techniques still need to be more effective and efficient against the network layer attack so that network layer attacks can be better detected in real-time. In our study, graphs are used to depict network flows, where the nodes represents devices, and the edge symbolizes the data flow between those devices. The graph representation of network traffic provides spatial features that is features related to how devices are connected inside working space of an organization. Anomalies in the graph representing network traffic are detected using temporal features when the graph changes unusually due to network flow over time. GNN has improved detection of patterns in graph structure, and LSTM captures time related features. Our paper deploys GNN-LSTM model to detect network layer attacks. The model used for detecting network layer attacks using GNN and LSTM extracts spatiotemporal features from network traffic. Deep Metric Learning (DML) is utilized to enhance the differentiation between the normal and attack traffic samples, and a Support Vector Machine (SVM) is used to classify the normal and attack network traffic segments. The performance of the model is evaluated against the existing machine learning model on publicly available network traffic data that contains information about normal and denial-of-service network packets.

In the upcoming sections, our paper will discuss anomaly attack detection, the creation of graphs from network traffic, and methods used to detect anomalies in graphs representing network traffic segments. Further, our paper concludes with the experimental results, analysis of the results, and future work. Our study contributes to the research field for detecting abnormalities in network traffic data.

II. Related work

Techniques for detecting network layer attacks can be categorized as detecting outliers, statistical approaches, machine learning techniques, temporal methods, and ensemble methods. Local Outlier Factor (LOF) [1], [2] that identifies instances that are outlier compared to other instances. Statistical approaches [3], [4] are also utilized for anomaly detection. Temporal methods [5], [6] are used to detect anomalies in temporal datasets. Unsupervised learning methods [2], [7]–[9] are approaches for anomaly detection in unlabeled datasets. Various studies involve autoencoders [10], [11] and generative adversarial networks [12] for anomaly detection for labeled datasets. Graph autoencoder is used for anomaly detection for unlabeled datasets [13]. Ensemble methods that include autoencoders with statistical methods [14] are used for anomaly detection. Autoencoder have been used with the LSTM for anomaly detection in network flow data [15]. In our study, we create graphs using network traffic segments and use GNN autoencoder with LSTM to capture graph structure, node features, and time variant features. Moreover, our study utilizes deep metric learning to decrease the distance between anomalous

samples, to solve the problem of multiple anomalous clusters. If there are multiple clusters for anomalous samples, and distance is decreased between samples, the anomalous samples can be grouped in one cluster so that classification between the normal and anomalous samples can be improved.

III. Dataset

The computer networking operations dataset from Visual Analytics Science and Technology (VAST) 2011 dataset [16] is used in our study. The computer networking operations dataset includes timestamp, source, and destination IP address. DOS attack started at timestamp 11:39:51; before the attack began at timestamp 11:39:51, the traffic was benign and posed no threat. After that time, the traffic is part of an attack and poses a threat. More information about the data can be found in [17].

A. Data processing

Figure 1 shows the creation of graphs every 5 seconds for two network traffic segments. The dotted line shows the separation of the network traffic segment into two segments. The top section in Figure 1 shows the traffic segment from time t_0 to time t_0+5, whereas the bottom portion of Figure 1 shows traffic segment from time t_0+5 to time t_0+10. Devices in the dataset are categorized into three categories based on their IP address and subnet mask as listed below:

- *Devices inside the organization* - Devices with IP addresses in the subnet 192.168.2.10–250 are internal devices in the organization since they are private IP addresses and are represented by a vector of [1,0,0] as node feature in a graph.
- *Webservers* - Devices with IP addresses of Firewall to the Internet, Firewall to EWS, External Web, Snort IDS to Network, Firewall to Data Center VLAN, Firewall to Office VLAN, DHCP, HR DB, Shipping/Routing DB, Internal Web, Mail Server, File Server, DNS, and Firewall Log are represented as webservers which provide services to devices. Webservers are denoted by a vector of [0,1,0] as a node feature in a graph.
- *Internet* - Devices with IP addresses in the subnet 200.150.1-255 are the devices outside the organization, because of the public IP address, and the devices outside the organization are represented by a vector of [0,0,1] as a node feature in a graph.

In Figures 1 and 2, devices 1, 2, and 6 have private IP addresses and are denoted by a vector of [1,0,0]. Similarly, devices 4 and 7 have public IP addresses so they are represented by a vector of [0,0,1]. Devices 3 and 5 are the servers and are depicted by a vector of [0,1,0].

Figure 1 demonstrates the two network traffic segments of 5-seconds intervals, and two graphs are created, where nodes are the unique IP addresses for the 5-second network traffic segment and edges are formed if there is data transfer between devices. The nodes are assigned feature vectors based on their IP addresses. Similarly, Figure 2 shows a construction of a graph from a non-graphical network traffic data segment

Packet Number	Time Stamp (Hour: Minute: Seconds)	Src IP	Dest IP
1	08:58:01	2	3
2	08:58:02	3	2
3	08:58:03	4	3
4	08:58:04	1	3
5	08:58:05	3	1
6	08:58:06	1	3
7	08:58:07	1	2
8	08:58:07	2	1
9	08:58:08	4	3
10	08:58:09	2	3
11	08:58:10	3	2

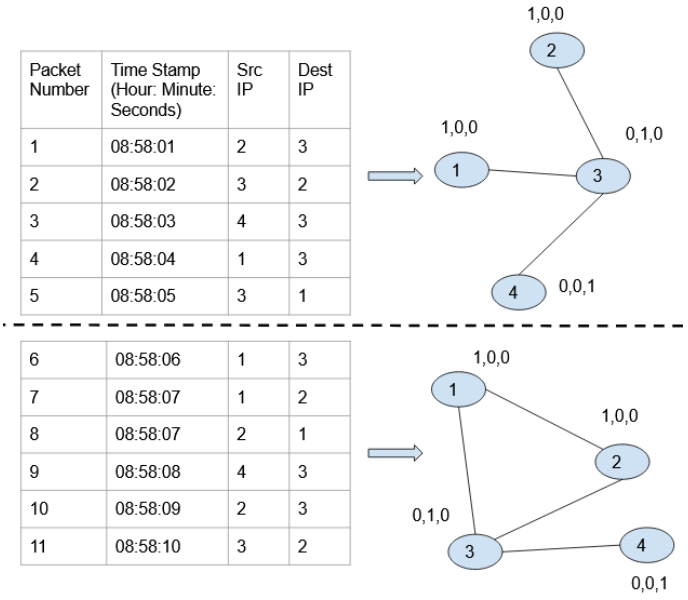

Fig. 1. Graph(5S) creation: Each graph is created with 5 seconds of network data. The dashed indicates that a graph is created for 5 seconds of network traffic and new graphs will be created continuously for every 5 seconds of network traffic.

Packet Number	Time Stamp (Hour: Minute: Seconds)	Src IP	Dest IP
1	08:25:01	2	3
2	08:25:02	5	7
3	08:25:03	7	5
4	08:25:04	5	6
5	08:25:05	6	1
6	08:25:06	4	3
7	08:25:06	1	3
8	08:25:07	3	2
9	08:25:08	6	5
10	08:25:09	1	6
11	08:25:10	3	4

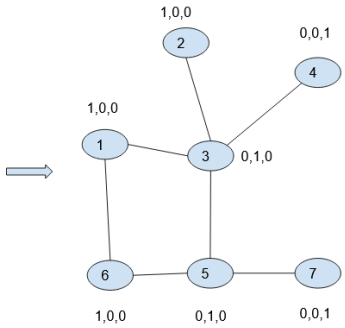

Fig. 2. Graph(10S) creation: A graph is created for 10 seconds of network traffic and new graph will be created continuously for every 10 seconds of network traffic.

of 10 seconds, where nodes are the devices with unique IP addresses and edges show the data transfer between the devices. Graphs(90s), Graphs(45s), Graphs(15s), Graphs(10s), Graph(8s), Graph(5s), and Graph(2.5s) datasets were created based on network traffic segments of 90, 45, 15, 10, 8, 5, or 2.5 seconds used for generating a graph in their respective graph dataset.

IV. Methodology

A. Problem statement

A graph representing a network traffic segment is denoted as $G_i = (D, T)$, where $1 \leq i \leq n$, D is the set of devices, and T is the set of traffic flow between devices D in a network traffic segment. The graph is characterized by a node feature matrix F, which has dimensions $\mathbb{R}^{|D| \times v}$, where each row f_d of dimension v represents the feature vector of a node $d \in D$. The task is to classify the graph $G_i = (D, T)$ into normal and attack network traffic data segments represented by the graph.

B. Graph Convolutional Network Convolution (GCNConv)

The GCNConv is used for feature extractor in our model by iteratively updating node embedding of graphs using embedding from neighboring nodes, where nodes belong to the graph that represent a network traffic segment. Given network traffic segment graph $G_i = (D, T)$, H^d_{p+1} denotes the embedding vector for node d at the $(p+1)^{th}$ layer, which is computed by using the embedding from the neighbors of the node d at p^{th} layer for every node $d \in D$. In formulation, H^d_{p+1} is calculated as follows [18]:

$$H^d_{p+1} = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H_pW_p) \quad (1)$$

In (1), $\tilde{A}$ is the adjacency matrix of graph G_i representing network traffic with self-loops. However, the graph G_i does not have self-loops. Self-loops are added to the adjacency matrix by adding identity matrix I_M to the adjacency matrix $\tilde{A} = A + I_M$. $\tilde{D}$ is a matrix containing the degree of devices connected in the network. W_p is the weight matrix for p^{th} layer, and $\sigma(.)$ is the sigmoid activation function. $H_p \in \mathbb{R}^{|D| \times h}$ is the hidden p^{th} layer with dimension h for every device $d \in D$, and $H_0 = F$.

C. Long Short-Term Memory (LSTM)

LSTM captures temporal dependencies for time-series data [19]. LSTM addresses the vanishing gradient problem that occurs in Recurrent Neural Network (RNN), while modeling long sequences of data [20]. So, LSTM was selected as one of our model components instead of RNN.

D. Triplet Margin Loss

Triplet Margin Loss [21] is a loss function that is used to train deep neural networks to learn a distance metric. Triplet Margin Loss utilizes anchor, positive, and negative samples. In our study, anchor and positive samples are the graphs that represent attack traffic segments, and the negative samples are the graphs that represent normal traffic segments. The objective of using the loss is to push the graphs representing attack network traffic segment closer in the embedding space, while pushing the graphs representing normal traffic segments away from the graphs representing attack traffic segments. The Triplet Margin Loss [22] is defined mathematically as:

$$TripletMarginLoss = max(d^k_{a,p} - d^k_{a,n} + m, 0) \quad (2)$$

In (2), a is the anchor sample, p is the positive sample, n is the negative sample, k is the number of samples, and m is the margin which is a hyperparameter that defines the minimum distance between the anchor and negative samples. $d^k_{a,p}$ is the distance between an anchor and a positive sample, and $d^k_{a,n}$ is the distance between an anchor and a negative sample [23].

E. Support Vector Machine (SVM)

SVM [24] is a supervised learning algorithm used for classification of graphs that represent normal and attack traffic segments. The SVM is utilized to find a hyperplane that maximally separates the graph samples to their respective classes. Sequential Minimal Optimization (SMO) [25] is used for the optimization of the hyperplane and support vectors in classifying the graphs.

F. Overview

GCNConv and LSTM work together to form an autoencoder for feature extraction. The GCN-LSTM model extracts both spatial and temporal features from the input graph data, which represent the connections between devices in the network. These features are then input into the deep metric learning algorithm to optimize the distances between anchor, positive, and negative samples. The optimized features are then provided to a SVM for classification, as shown in Figure 3. The GCN-LSTM combination allows for the extraction of spatiotemporal features, capturing both the structure and the temporal details related to the graphs representing network traffic segments.

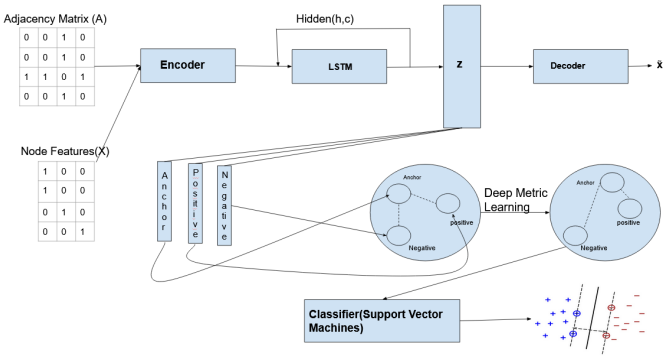

Fig. 3. Model for feature extraction and classification.

Figure 4 shows the normal graph structure of the network and the abnormal graph structure. Our model will learn the graphical pattern for the normal graph and abnormal graph from the training and look for the pattern in the test data. The classification of benign and malicious traffic is based on the assumption that there is a distinct spatiotemporal pattern in the graph. In Figure 4 a node with features [0,0,1], representing an external device, being directly connected to a node [1,0,0], representing an internal device in an organization, is considered an abnormal connection and the resulting graph is abnormal graph structure. In the normal graph structure, the internal devices communicate with each other and webservers, whereas the Internet devices communicate with the webservers.

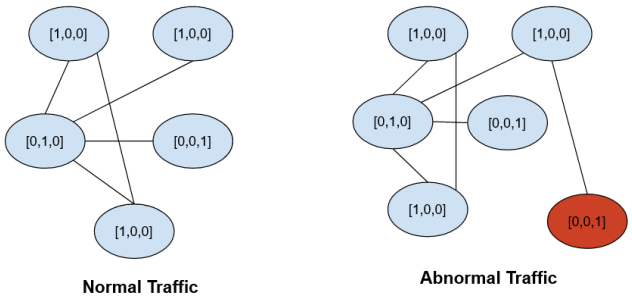

Fig. 4. Normal pattern is shown by normal traffic graph structure, and Abnormal pattern is shown by abnormal traffic graph structure.

The LSTM component in the model helps to understand the temporal aspect of the connections, such as the expected initiation of communication is from the internal device to webservers and then the connection is initiated by the webserver device to the internet devices. The LSTM learns these temporal patterns from the training data to enable the accurate classification of normal and abnormal traffic on test data.

V. Experiments

The proposed model, along with other models, and the existing Graph Based Anomaly Detection (GBAD) model was evaluated on seven different datasets and a comparison of performance was made on these datasets.

A. Experimental setup

In Table I, the GCN-LSTM model consists of an encoder, decoder and a linear layer that captures the latent feature of a graph by learning information related to graph structure, node features of a graph, and sequential patterns between graphs. The GCN-LSTM encoder layer comprises of two GCNConv layers and an LSTM layer that can process a graph of 164 nodes and three features associated with each node and produce latent features as their output. The latent features serve as the input to GCN-LSTM decoder layers, which consist of two GCNConv layers and a linear layer. The purpose of GCN-LSTM decoder layers is to reconstruct the input graph from the latent characteristics of the graph, which produces final output of 164 nodes with each node having three features.

TABLE I
Our model architecture (IFL = Input Feature Length, OFL = Output Feature Length).

Model	Architecture
GCN-LSTM	Encoder: $GCNConv(3, 16)$ Encoder: $GCNConv(16, 16)$ Encoder: $LSTM(16, 16)$ Decoder: $GCNConv(16, 16)$ Decoder: $GCNConv(16, 3)$ Decoder: $Linear(IFL = 3, OFL = 3, bias = True)$
DML	$Linear(IFL = 2624, OFL = 64, bias = True)$ $Linear(IFL = 64, OFL = 2624, bias = True)$
SVM	$Kernel =' linear'$

The reconstruction loss is computed between the input graph and final output graph of GCN-LSTM model using Negative Log-Likelihood Loss (NllLoss) function. The loss is minimized using an Adaptive Moment Estimation (ADAM) [26] optimizer with a learning rate of 0.001 and a weight decay of 0.0005. The learning rate dictates the direction and magnitude of the weight updates made by the optimizer during each iteration, and the weight decay is a regularization parameter that penalizes the larger weights.

DML model consists of two linear layers, where a triplet margin loss function with a margin value of 0.5 is used to reduce the distance within positive or negative class and increase the distance between positive and negative class. The loss in the DML model is minimized using an ADAM optimizer with a learning rate of 0.001 and a weight decay of 0.0005. Finally, the output of the DML model is passed as input for a SVM, that uses a linear kernel and has a cost parameter value of 10. The optimization of the SVM is conducted using Sequential Minimal Optimization to perform the classification task.

VI. Results

The model is trained and evaluated on datasets Graph(90s), Graph(45s), Graph(15s), Graph(10s), Graph(8s), Graph(5s), and Graph(2.5s). Each dataset is separated into two subsets for training and testing. The training subset contains 80% of the dataset, and the testing subset contains 20% of the dataset. The training subset is used to train the model, whereas the testing subset is used to evaluate and compare the performance of models using accuracy, precision, recall, and F1 score metrics and the results are reported in Table II. The training of the model is performed for 100 epochs, with Xavier initialization [27] applied to initialize the weights and parameters.

TABLE II
Our model's performance was evaluated on the testing sets of four distinct datasets.

Graphs	Accuracy	Precision	Recall	F1 score
`Graph(90s)`	0.95	1.00	0.95	0.97
`Graph(45s)`	0.97	0.97	1.00	0.98
`Graphs(15s)`	0.99	0.99	1.00	0.99
`Graph(10s)`	0.99	1.00	0.99	0.99

GBAD (Graph Based Anomaly Detection) [17] model uses the Minimum Description Length (MDL) that minimizes the description length of graph using the normative substructure S [28]. An anomalous graph substructure S' is detected in graphs that are not isomorphic to the graph's normative substructure S.

In the Hierarchical clustering [29] model, Hierarchical clustering is applied to the similarity matrix computed by comparing graphs degree distribution using the Kolmogorov-Smirnov test. After Hierarchical clustering, the graphs are grouped using the similarity matrix.

In the spectral moments [30] model, the Laplacian matrix of each graph is computed, and the Laplacian matrix is decomposed into eigenvalues and eigenvectors. Based on the

eigenvalues, the spectral moment of the graph is computed, which captures the properties of the graph structure. The computed spectral moments of graphs are used to cluster the graph using the K-means algorithm.

In the Graph Isomorphism Network Convolution (GINCONV) [31] model, GINCONV layers are used to form an autoencoder that extracts latent features from graphs and the latent features are then used to cluster graphs by utilizing the k-means algorithm.

The results of our model for the Graph(5s) dataset are compared to GBAD models, Hierarchical Clustering (HC), spectral moments, and GINConv in Table III.

TABLE III
COMPARISON WITH OTHER MODEL USING GRAPH(5S) DATASET.

Models	Accuracy	Precision	Recall	F1 score
GBAD model	0.99	1.00	0.96	0.98
HC	0.38	1.00	0.34	0.51
Spectral moments	0.88	1.00	0.87	0.93
GINCONV	0.52	1.00	0.49	0.66
Our Model	0.99	1.00	0.99	0.99

Our model performed better than HC, spectral moments, and GINCONV. Our model and GBAD model both have comparable performance on the Graph(5s) dataset, further comparisons are made using Graph(8s) and Graph(2.5s) datasets. The percentage of anomaly graphs reported by our model is compared to the GBAD model on Graph(8s), and Graph(2.5s) and are reported in Table IV. [17] has reported the percentage of anomaly graphs detected by the GBAD model on Graph(8s) and Graph(2.5s). So, the comparison between the GBAD model and our model is made on Graph(8s) and Graph(2.5s) using the percentage of anomaly graph reported. Our model performed better than GBAD model in both Graph(8s) and Graph(2.5s) datasets.

TABLE IV
PERFORMANCE OF GBAD AND OUR MODEL ON ANOMALY GRAPH REPORTED.

Graphs	GBAD Model	Our model
Graph(8s)	1.35 %	100 %
Graph(2.5S)	18.40 %	100 %

Table V displays the variances of the attack graphs before and after applying deep metric learning. The values of agvar(B) correspond to variance of graphs that represent attack traffic segments $before$ applying deep metric learning, while the values for agvar(A) represent the variance of graphs representing attack traffic segments $after$ applying deep metric learning.

Deep metric learning decreases the distance within classes, which reduces the variance within classes. Lowering variance within class results in better classification accuracy.

The NllLoss while reconstructing the node features of the Graph(5s) dataset while extracting features is reported in Figure 5. The NllLoss measures the dissimilarity between the predicted node features by the GCN-LSTM model and the input graph node features.

TABLE V
VARIANCE OF ATTACK CLUSTERS AND NORMAL CLUSTERS BEFORE AND AFTER USING TRIPLET MARGIN LOSS.

Graphs	agvar(B)	agvar(A)
Graph(90s)	3.13×10^{-6}	7.10×10^{-8}
Graph(45s)	2.30×10^{-7}	1.80×10^{-9}
Graph(15s)	4.95×10^{-7}	1.49×10^{-9}
Graph(10s)	5.35×10^{-7}	3.28×10^{-8}
Graph(8s)	6.35×10^{-5}	1.11×10^{-6}
Graph(5s)	9.69×10^{-7}	5.06×10^{-9}
Graph(2.5s)	6.52×10^{-6}	5.27×10^{-7}

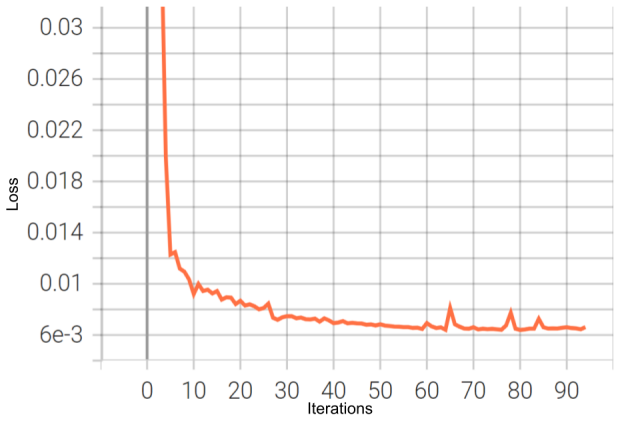

Fig. 5. Loss on training set on the graph(5s) dataset.

The accuracy during reconstruction of the node features of the graph in Graph(5s) dataset is reported in Figure 6. The plot shows the model performance on predicting the node features of graphs, which shows the model ability to learn the underlying structure and characteristics of graphs. Accuracy of reconstructing the node feature will have an influence on the performance of the model in detecting anomalies. Therefore, it is important to evaluate both the node feature reconstruction

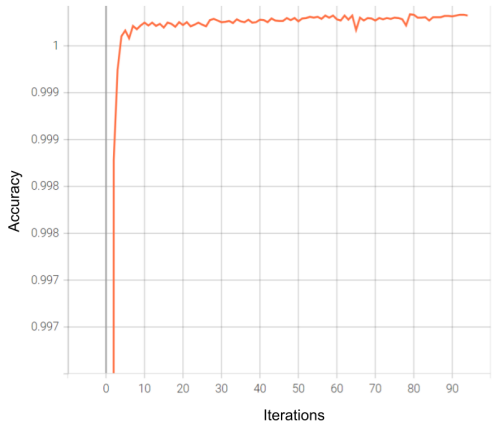

Fig. 6. Accuracy on training set on the Graph(5s) dataset.

accuracy and the anomaly detection performance to show the effectiveness of our model in detecting anomalies in graphs.

TABLE VI
RUNTIME COMPLEXITY OF GRAPH CREATION, NUMBER OF GRAPHS CREATED (N_G), TRAINING, AND TESTING TIME OF OUR MODEL ON SEVEN DATASETS ARE REPORTED.

Graphs	Creation (s)	N_G	Training (s)	Testing (s)
Graph(90s)	32.20	118	141.71	0.36
Graph(45s)	32.95	231	252.31	0.49
Graphs(15s)	33.87	651	701.36	0.78
Graph(10s)	34.19	944	934.71	1.28
Graph(8s)	34.59	1140	1228.65	1.40
Graph(5s)	36.29	1653	1852.77	1.90
Graph(2.5s)	40.06	3201	3433.37	3.41

In Table VI, the time required in seconds to create the graph from the VAST network traffic dataset which contains the network traffic data from 8:52:52 to 11:46:34 is reported. Training time refers to the time required in seconds for the model to train on the training subset for 100 epochs whereas testing time refers to the time required in seconds for the model to classify the graphs as normal or abnormal graph in the testing subset. As the time interval of the network traffic segment used to create graphs decreases from 90 seconds to 2.5 seconds, the number of graphs (N_G) created from the dataset increases since shorter interval creates more network traffic segments. As the number of graphs increases due to decreasing time intervals for creating graphs, graph creation, training, and testing time of our model on the training subset and testing subset of graph datasets from Graph(90s) to Graph(2.5s) increases.

VII. DISCUSSION

Our GNN-based model performs better than machine learning models like HC, spectral moments, and GINCONV on the graph(5s) dataset. Our model has a higher accuracy in detecting anomalies than the GBAD model, as shown in Table IV. Our GNN-based model uses deep metric learning, which decreases variance within a class, as shown in Table V. The decrease in the variance is crucial for classification accuracy. The deep learning algorithm is also able to address the issue where there are multiple clusters of anomaly samples having different properties by grouping them together by reducing the variance within anomaly class. Deep metric learning is able to decrease the distance within classes by projecting them in a different dimension and using the triplet margin loss function to guide the projection. Therefore, our model will perform better than other models when there are multiple clusters of anomaly samples. Deep metric learning decreases the distance between anomaly samples, and as the distance between the anomaly samples is reduced, that reduces the variance within the anomaly class, and as a result, our model shows higher classification accuracy. Our model can have better or similar performance when there is a single cluster of anomaly samples in the dataset. Moreover, HC, spectral moments, GINCONV, and GBAD models focus on the structure of the graphs, and do not focus on the change in the graph structure over time. Our model captures the temporal relationship between segments of data packets sent and received in network traffic. Additionally, SVM is a better classifier with several graphs as reported in Table VI with high dimensional features that represent the graph, therefore, our model uses SVM as a classifier. As shown in Figure 5, the loss decreases throughout training process, which indicates model ability to learn useful features of graphs. It can be observed in Figure 6 that the accuracy of our GNN-based model increases as the training progresses.

A. Conclusion

In conclusion, our study demonstrates the effectiveness of GNN in detecting an anomaly in network traffic. The graphs are visualizations of devices and their connections within the network that help in determining the network topology, which acts like space related features within an organization, also called spatial features. Each graph in the graph dataset changes over time, which captures the time related features, also known as temporal features. The graphs are made by using interaction between devices in network traffic segments. Our GNN-based model outperforms other machine learning techniques and achieves higher accuracy in the seven different datasets for detecting anomalies in the network. Our study suggests that GNN-based model has the potential to improve anomaly detection and contribute to network layer attack detection. Our study also revealed that deep metric learning could reduce the distance and variance between samples within the class in the dataset.

However, in our model, the LSTM does not capture the time delay between each network traffic packet within a network traffic segment, which leads the model to capture the features that do not include information of time delay and could not detect the anomalies caused by time delay. In future work, the time delay between network packets will be incorporated as an edge feature that will capture information about time delay in the extracted features, which will detect anomalous network traffic segments due to time delays in network traffic. Node features can be used to incorporate more information about devices in the graphs, which will enhance the information content of the nodes in the graphs. In the future, more features can be added to the node by capturing network traffic that contains more information about the devices that are in the network traffic data.

REFERENCES

[1] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," *ACM SIGMOD Record*, vol. 29, no. 2, pp. 93–104, May 2000.

[2] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in *2008 Eighth IEEE International Conference on Data Mining*, Dec. 2008, pp. 413–422, iSSN: 2374-8486.

[3] R. McGill, J. W. Tukey, and W. A. Larsen, "Variations of Box Plots," *The American Statistician*, vol. 32, no. 1, pp. 12–16, 1978, publisher: [American Statistical Association, Taylor & Francis, Ltd.].

[4] G. Mclachlan, "Mahalanobis Distance," *Resonance*, vol. 4, pp. 20–26, Jun. 1999.

[5] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*. Holden-Day, 1976, google-Books-ID: 1WVHAAAAMAAJ.

[6] S. J. Taylor and B. Letham, "Forecasting at Scale," *The American Statistician*, vol. 72, no. 1, pp. 37–45, Jan. 2018.

[7] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1967.

[8] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the Support of a High-Dimensional Distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, Jul. 2001. [Online]. Available: https://direct.mit.edu/neco/article/13/7/1443-1471/6529

[9] S. Guha, N. Mishra, G. Roy, and O. Schrijvers, "Robust random cut forest based anomaly detection on streams," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. New York, NY, USA: JMLR.org, Jun. 2016, pp. 2712–2721.

[10] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, "Autoencoder-based network anomaly detection," in *2018 Wireless Telecommunications Symposium (WTS)*, Apr. 2018, pp. 1–5.

[11] D. Zhu, Y. Ma, and Y. Liu, "Anomaly Detection with Deep Graph Autoencoders on Attributed Networks," in *2020 IEEE Symposium on Computers and Communications (ISCC)*, Jul. 2020, pp. 1–6, iSSN: 2642-7389.

[12] X. Xia, X. Pan, N. Li, X. He, L. Ma, X. Zhang, and N. Ding, "GAN-based anomaly detection: A review," *Neurocomputing*, vol. 493, pp. 497–535, Jul. 2022.

[13] M. Shao, Y. Lin, Q. Peng, J. Zhao, Z. Pei, and Y. Sun, "Learning graph deep autoencoder for anomaly detection in multi-attributed networks," *Knowledge-Based Systems*, vol. 260, p. 110084, Jan. 2023.

[14] C. Ieracitano, A. Adeel, F. C. Morabito, and A. Hussain, "A novel statistical analysis and autoencoder driven intelligent intrusion detection approach," *Neurocomputing*, vol. 387, pp. 51–62, Apr. 2020.

[15] Y. Zhong, W. Chen, Z. Wang, Y. Chen, K. Wang, Y. Li, X. Yin, X. Shi, J. Yang, and K. Li, "HELAD: A novel network anomaly detection model based on heterogeneous ensemble learning," *Computer Networks*, vol. 169, p. 107049, Mar. 2020.

[16] IEEE VAST Challenge Committee, "Ieee vast challenge 2011 dataset," IEEE Visual Analytics Science and Technology Challenge, 2011, Computer Networking Operations. [Online]. Available: https://visualdata.wustl.edu/varepository/benchmarks.php

[17] R. Paudel, P. Harlan, and W. Eberle, "Detecting the Onset of a Network Layer DoS Attack with a Graph-Based Approach," in *The thirty-second international flairs conference*, May 2019.

[18] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *International Conference on Learning Representations*, Feb. 2017.

[19] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, conference Name: Neural Computation.

[20] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, ser. ICML '06. New York, NY, USA: Association for Computing Machinery, Jun. 2006, pp. 369–376.

[21] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 815–823, iSSN: 1063-6919.

[22] G. Kasif and G. Thondilege, "Exploring Music Similarity through Siamese CNNs using Triplet Loss on Music Samples," in *2023 International Research Conference on Smart Computing and Systems Engineering (SCSE)*, vol. 6, Jun. 2023, pp. 1–8, iSSN: 2613-8662. [Online]. Available: https://ieeexplore.ieee.org/document/10215020

[23] Y. Han, H. Zhu, L. Jiao, X. Yi, X. Li, B. Hou, W. Ma, and S. Wang, "SSMU-Net: A Style Separation and Mode Unification Network for Multimodal Remote Sensing Image Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–15, 2023, conference Name: IEEE Transactions on Geoscience and Remote Sensing. [Online]. Available: https://ieeexplore.ieee.org/document/10271328

[24] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[25] J. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines," *Advances in Kernel Methods-Support Vector Learning*, vol. 208, Jul. 1998.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, 2014.

[27] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Journal of Machine Learning Research - Proceedings Track*, vol. 9, pp. 249–256, Jan. 2010.

[28] W. Eberle and L. Holder, "Anomaly detection in data represented as graphs," *Intelligent Data Analysis*, vol. 11, no. 6, pp. 663–689, Nov. 2007.

[29] N. Ishaq, T. J. Howard, and N. M. Daniels, "Clustered Hierarchical Anomaly and Outlier Detection Algorithms," in *2021 IEEE International Conference on Big Data (Big Data)*, Dec. 2021, pp. 5163–5174.

[30] H. E. Egilmez and A. Ortega, "Spectral anomaly detection using graph-based filtering for wireless sensor networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 1085–1089, iSSN: 2379-190X.

[31] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "HOW POWERFUL ARE GRAPH NEURAL NETWORKS?" in *International Conference on Learning Representations*, 2019.