Graph Neural Networks for Graph-Level Anomaly Detection in Network Intrusion Detection

Georgios Anyfantis
Universitat Politècnica de Catalunya
georgios.anyfantis@upc.edu

Ismael Castell-Uroz
Universitat Politècnica de Catalunya
ismael.castell@upc.edu

Pere Barlet-Ros
Universitat Politècnica de Catalunya
pere.barlet@upc.edu

***Abstract*—Network attacks are a growing problem in today's world. Their number is growing exponentially, and they become more sophisticated every year, accentuating the need for robust Network Intrusion Detection Systems. Graph Neural Networks have been proven to be very capable in detecting and categorising network attacks, making them ideal for detection in real-life deployments. However, they are limited to training on a small number of outdated labelled datasets. We propose a new method for anomaly detection through a graph heterogeneous autoencoder as a way to reduce reliance on labelled datasets. Our graph autoencoder reconstructs the structure and features of a graph representing the traffic connections in a network. We train the autoencoder with normal graphs and then derive embeddings to use with traditional anomaly detection based on Machine Learning. Our results indicate that our solution can create distinct normal and anomalous graph embeddings that can then be used to detect attacks with a high degree of accuracy.**

I. Introduction

In recent years, attacks and intrusions have been a growing problem. The number of attacks has grown exponentially, with attacks becoming more and more sophisticated [1]. As such, a need for robust Network Intrusion Detection Systems (NIDS) has arisen.

Quite a lot of research has been conducted on the application of machine learning methods for the detection and characterisation of attacks [2]–[5]. However, many of these approaches are susceptible to adversarial attacks, making them unsuitable for real-life deployment [6]. Graph Neural Networks have been proven to be resistant to adversarial attacks as they can learn about the characteristics and structures of the graph [7]–[9]. Thus, they are an ideal choice for real-life deployment.

Another issue that arises in the domain of NIDS, especially in the domain of machine learning approaches, is that the datasets that are currently available for the creation of those models are limited [10]. Many of these datasets contain outdated attacks that are no longer in use and created a distorted picture of how attacks are [11]. This in turn creates models that are not representational of real-life deployment. In addition, some data sets have been found to have incorrect labels, making their use problematic as models do not learn to correctly recognise attack patterns [11].

Anomaly detection is a good approach to mitigate reliance on labelled datasets, as we only need normal data to establish a baseline, and anything that deviates is considered anomalous.

978-3-903176-74-4

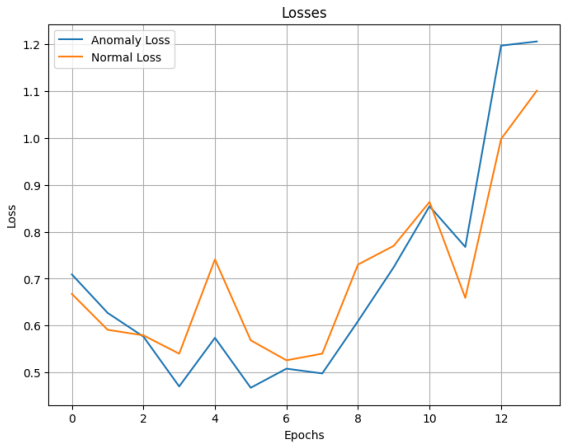

Fig. 1. Anomalous and Normal Graph Reconstruct Errors during Validation

In the context of GNN for NIDS, Anomal-E has been proposed [12]. Anomal-E works by creating a homogeneous graph, with network flows being edge features that are then used to create edge embeddings that are then used for anomaly detection.

Another approach is to perform graph-level anomaly detection. Graph-level anomaly detection works by trying to detect if a graph is anomalous or not [13]. Many approaches rely on the use of autoencoders to calculate the reconstruction errors, which are then used to detect anomalous graphs. However, this approach is problematic due to the reconstruction flip, a scenario where the anomalous graph reconstruction error is similar to or better than normal graphs, which renders anomaly detection very difficult. This can be observed in Figure 1 during the training of our graph autoencoder (described later in Section II), where the reconstruction error of anomalous graphs is very similar to that of normal graphs, and even decreases below it after epoch 2, which is consistent with recent findings in [13].

We propose the use of graph-level anomaly detection. We are mainly interested in capturing normal data that is then used to train our Graph Autoencoder (GAE). Our autoencoder is trained to reconstruct both the features and the structure of the graphs to a very high degree. Instead of using the reconstruction error for anomaly detection, our autoencoder is capable of mapping normal graphs into the latent space, with anomalous graphs mapping further away than normal

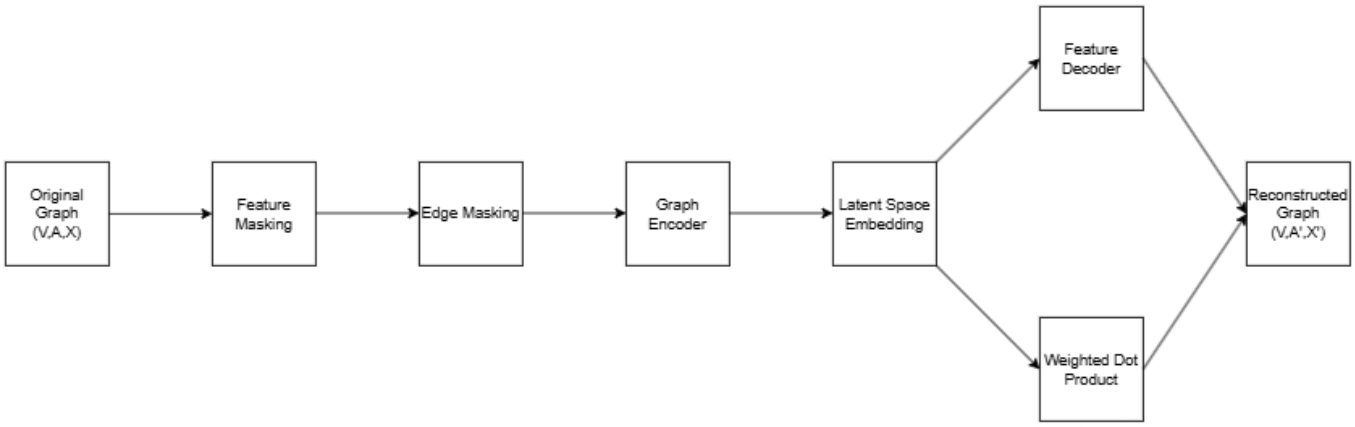

Fig. 2. Our Autoencoder Design

graphs, allowing for their detection. Our approach indicates the creation of unique embedding clusters between normal and anomalous graphs, making their detection using traditional machine learning approaches very accurate. This approach mitigates the reconstruction flip that other autoencoder approaches suffer from (i.e., [13]).

II. Methodology

Our autoencoder design has been inspired by the Graph Masked Autoencoder (GraphMAE) [14] and the Heterogeneous Graph Masked Autoencoder (HGMAE) [15]. We employ feature masking and edge dropping during training to reduce the reliance of the autoencoder on trivial solutions and force it to learn the relationships between the nodes of the graph. The masking used for the graph features uses a learnable masking token similar to BERT [16], but only the features in the input graph are masked and not the latent space, as is the case in GraphMAE. The autoencoder design can be seen in Figure 2.

A. Feature and Structure Reconstruction

The main difference between our approach and GraphMAE and HGMAE is that those approaches rely on Cosine Error for their feature reconstruction while we rely on Mean Square Error (MSE) as we need the features to be as close as possible to the original ones due to the nature of our task. Furthermore, both approaches only reconstruct the masked features, while our approach reconstructs all the features that we are interested in reconstructing. The importance of structural and feature reconstruction in the final reconstruction error is regulated by user-defined weights.

When it comes to structural reconstruction, we employ a weighted Dot Product. Heterogeneous graphs can be split into smaller bipartite graphs, which simplifies the structure of the graph [17]. The way our weighted Dot Product works is that we rebuild the adjacency matrix for each of the relationships between the nodes using Dot Product like in the original GAE paper [18]. The main difference is that we use learnable weights to better capture the inter-node relationships. As such, we can better capture the graph structure.

B. Dataset and Graph Creation

We used the UNSW-NB15 dataset [19] to train the autoencoder and perform anomaly detection. We chose to use UNSW-NB15 because it is one of the most used datasets in the area that contains clearly labelled modern attacks. Our graphs are structured in a way that creates separate IP and Flow nodes that are connected. That allows us to capture more information on how anomalous graphs are structured as different attacks result in different attack structures [7].

For graph creation, we first start by cleaning the dataset and removing any missing or infinite values. Following that, we split the dataset into the dates that make up it. We chose the first and second dates for training and validation, while the third was kept as a testing portion. Following the splits, we perform data normalisation. We employ a Yeo-Johnson Power Transformation [20] to bring the data closer to a normal distribution followed by Min-Max Scaling to ensure a fixed scale of the dataset. Finally, we create the graphs using a sliding-window technique. The windows that we used have a size of 160 seconds and the algorithm used a stride of 160 seconds, ensuring that there are no overlapping windows. We used scikit-learn [21] for data normalisation and scaling.

C. Embedding Creation and Anomaly Creation

For the creation of the graph embeddings, we have chosen first to create the node embeddings and then combine them. We first pool each node type on their own and then combine them to a single vector embedding that represents the entire graph. We have created multiple different approaches to create these embeddings. For calculating the embeddings for each node, we can calculate the mean, max, and sum of all nodes. As for the graph embedding, we combine the embeddings of the nodes, and we either concatenate them, calculate their average, or add all of them together.

For anomaly detection, we rely on traditional machine learning approaches, such as One-class SVM [22], Isolation

Forest [23], Local Outlier Factor (LOF) [24], KMeans clusters [25], DBSCAN [26] and Gaussian Mixture Model (GMM) [27]. We implement our GNN using PyTorch Geometric [28] and PyTorch Lightning [29]. For traditional ML approaches using scikit-learn [21].

D. Embedding Hyperparameters

To train the autoencoder, we chose the following hyperparameters and GNN algorithms. For the Encoder and Decoder, we chose to use GATv2 [30]. GATv2 was chosen because it is an improvement over the original GAT architecture by employing a dynamic attention mechanism, making it more expressive and producing better results than the algorithm paper. We used ReLU as the activation function in both the encoder and decoder with Weighted Dot Product for the structural reconstruction. We used a three-layer architecture on both the encoder and the decoder with the following dimensions [16, 32, 32]. Finally, we used five attention heads and concatenation resulting in node embeddings of length 160.

III. Results

To choose the best combination for anomaly detection, we created the embeddings for every possible combination and then calculated metrics to identify the combination that creates distinct clusters. The metrics we chose are the Silhouette score, the Bavies-Bouldin (BD) index, and the Calinki-Harabasz (CH) index. The silhouette score [31] measures how well a point is within its group, with a higher score being better. BD index computes the average ratio between within-cluster scatter and between-cluster separation for each cluster, with lower scores being tighter clusters.Finally, the CH index is the ratio of the dispersion between clusters to within clusters, where a higher value indicates well-separated clusters.

Table I shows different combinations of the pooling and node schemes. The best performance achieved is from the mean node grouping with the node pooling concatenation. They achieve the best separation between clusters, with the embeddings being very close to their own cluster and that all clusters are distinct and compact. Consequently, we selected this combination to perform the rest of the experiments.

Using the embeddings produced by the chosen combination, Node Concatenation and Mean Pooling, we trained the anomaly detection algorithms on embeddings from normal graphs that were used in the training of the GAE. We then used the remaining part of the dataset to test how well traditional approaches can detect anomalies. The test dataset has 44 normal graphs and 244 anomalous graphs. Tables II and III show the hyperparameterization and the results obtained.

Our experiments in Table III show very good results overall with many algorithms with accuracy greater than 96%. Thus, they perform exceptionally well in classifying the anomalous graphs from the normal ones. The only ones that are not performing well are DBSCAN and GMM. In the case of DBSCAN, this could be due to the high dimensionality of the embeddings, but also due to the model not being able to detect anomalies if they are not clearly sparse [32]. For GMM, it could be the case that the data distribution is non-Gaussian, which could be causing the model to not be able to detect the anomalies or that the vector dimensionality is very high, resulting in GMM not being able to assign a low likelihood to true anomalies [33].

TABLE I
Clustering-validation metrics for different pooling and node–combination schemes (3 clusters).

Pooling	Grouping	Silhouette	DB Index	CH Index
Max	Concat	0.7674	0.4344	321.7952
Max	Sum	0.7790	0.4092	494.0815
Max	Average	0.7790	0.4092	494.0814
Mean	**Concat**	**0.8654**	**0.2908**	**1400.1959**
Mean	Sum	0.7548	0.2922	296.6385
Mean	Average	0.7548	0.2922	296.6384
Sum	Concat	0.8060	0.3372	609.3115
Sum	Sum	0.8060	0.3372	609.2721
Sum	Average	0.8060	0.3372	609.2722

TABLE II
Thresholds and key hyperparameters used by each method.

Method	Hyperparameters
One-Class SVM	threshold = 0.44
Isolation Forest	contamination = 0.01
LOF Encoding	contamination = 0.05
K-Means Clusters	10 clusters (distance th. = 95th percentile)
DBSCAN	$\epsilon = 0.5$, min_samples=5
Gaussian Mixture Model	$n\_components = 5$

TABLE III
Anomaly detection performance on validation set (Mean + Concat embeddings).

Method	Accuracy	Precision	Recall	AUROC
One-Class SVM	0.9381	0.9751	0.9514	0.9352
Isolation Forest	0.9416	0.9423	0.9919	0.9137
LOF Encoding	0.9519	0.9498	**0.9960**	0.9153
K-Means Clusters	**0.9588**	**0.9608**	0.9919	**0.9614**
DBSCAN	0.4089	0.7698	0.4332	0.7085
Gaussian Mixture Model	0.1031	0.0625	0.0040	0.8148

IV. Conclusion

In conclusion, throughout this extended abstract we have presented our approach to detect anomalous flows using GAEs and traditional Machine Learning anomaly detection approaches. We have showcased the excellent results from our autoencoder, which is able to create unique clusters that can then be used very successfully to detect anomalous graphs. Our approach mitigates the issues of reconstruction flip and yields excellent results in anomaly detection. Furthermore, training the autoencoder on graph reconstruction removes the need for labelled datasets that other algorithms rely on.

Acknowledgements

This work was supported by the CHISTERA grant CHIST-ERA-22-SPiDDS-02 corresponding to the GRAPHS4SEC project (reference nº PCI2023-145974-2) funded by the Agencia Estatal de Investigación through the PCI 2023 call. This work is also supported by the Catalan Institution for Research and Advanced Studies (ICREA Academia).

REFERENCES

[1] S. Kumar, S. Gupta, and S. Arora, "Research trends in network-based intrusion detection systems: A review," *IEEE Access*, vol. 9, pp. 157 761–157 779, 2021.

[2] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.

[3] S. Zaman and F. Karray, "Lightweight ids based on features selection and ids classification scheme," in *2009 International Conference on Computational Science and Engineering*, vol. 3, 2009, pp. 365–370.

[4] B. Babu, G. Reddy, D. Goud, K. Naveen, and K. T. Reddy, "Network intrusion detection using machine learning algorithms," in *2023 3rd International Conference on Smart Data Intelligence (ICSMDI)*, 2023, pp. 367–371.

[5] R. Chapaneri and S. Shah, "A comprehensive survey of machine learning-based network intrusion detection," in *Smart Intelligent Computing and Applications*, S. C. Satapathy, V. Bhateja, and S. Das, Eds. Singapore: Springer Singapore, 2019, pp. 345–356.

[6] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016, pp. 372–387.

[7] D. Pujol-Perich, J. Suárez-Varela, A. Cabellos-Aparicio, and P. Barlet-Ros, "Unveiling the potential of graph neural networks for robust intrusion detection," 2021. [Online]. Available: https://arxiv.org/abs/2107.14756

[8] T. Bilot, N. E. Madhoun, K. A. Agha, and A. Zouaoui, "Graph neural networks for intrusion detection: A survey," *IEEE Access*, vol. 11, pp. 49 114–49 139, 2023.

[9] Z. Sun, A. M. Teixeira, and S. Toor, "Gnn-ids: Graph neural network based intrusion detection system," in *Proceedings of the 19th International Conference on Availability, Reliability and Security*, ser. ARES '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3664476.3664515

[10] P. Goldschmidt and D. Chudá, "Network intrusion datasets: A survey, limitations, and recommendations," 2025. [Online]. Available: https://arxiv.org/abs/2502.06688

[11] D. Pinto, I. Amorim, E. Maia, and I. Praça, "A review on intrusion detection datasets: tools, processes, and features," *Computer Networks*, vol. 262, p. 111177, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128625001458

[12] E. Caville, W. W. Lo, S. Layeghy, and M. Portmann, "Anomal-e: A self-supervised network intrusion detection system based on graph neural networks," *Knowledge-Based Systems*, vol. 258, p. 110030, Dec. 2022. [Online]. Available: http://dx.doi.org/10.1016/j.knosys.2022.110030

[13] S. Kim, S. Y. Lee, F. Bu, S. Kang, K. Kim, J. Yoo, and K. Shin, "Rethinking reconstruction-based graph-level anomaly detection: Limitations and a simple remedy," 2024. [Online]. Available: https://arxiv.org/abs/2410.20366

[14] Z. Hou, X. Liu, Y. Cen, Y. Dong, H. Yang, C. Wang, and J. Tang, "Graphmae: Self-supervised masked graph autoencoders," 2022. [Online]. Available: https://arxiv.org/abs/2205.10803

[15] Y. Tian, K. Dong, C. Zhang, C. Zhang, and N. V. Chawla, "Heterogeneous graph masked autoencoders," 2023. [Online]. Available: https://arxiv.org/abs/2208.09957

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: https://arxiv.org/abs/1810.04805

[17] C. Shi, X. Wang, and P. S. Yu, *The State-of-the-Art of Heterogeneous Graph Representation*. Singapore: Springer Singapore, 2022, pp. 9–25. [Online]. Available: https://doi.org/10.1007/978-981-16-6166-2_2

[18] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016. [Online]. Available: https://arxiv.org/abs/1611.07308

[19] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6.

[20] I. Yeo and R. A. Johnson, "A new family of power transformations to improve normality or symmetry," *Biometrika*, vol. 87, no. 4, pp. 954–959, 12 2000. [Online]. Available: https://doi.org/10.1093/biomet/87.4.954

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[22] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller, Eds., vol. 12. MIT Press, 1999. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1999/file/8725fb777f25776ffa9076e44fcfd776\protect\discretionary{\char\hyphenchar\font}{}{}Paper.pdf

[23] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422.

[24] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," *SIGMOD Rec.*, vol. 29, no. 2, p. 93–104, May 2000. [Online]. Available: https://doi.org/10.1145/335191.335388

[25] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognition*, vol. 36, no. 2, pp. 451–461, 2003, biometrics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320302000602

[26] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*, vol. 96, no. 34, 1996, pp. 226–231.

[27] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977. [Online]. Available: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1977.tb01600.x

[28] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," 2019. [Online]. Available: https://arxiv.org/abs/1903.02428

[29] W. Falcon and The PyTorch Lightning team, "Pytorch lightning," March 2019. [Online]. Available: https://github.com/Lightning-AI/lightning

[30] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" *CoRR*, vol. abs/2105.14491, 2021. [Online]. Available: https://arxiv.org/abs/2105.14491

[31] P.-N. Tan, M. Steinbach, and V. Kumar, "Cluster analysis: Basic concepts and algorithms," in *Introduction to Data Mining*, 2nd ed. Harlow, Essex, England: Pearson, 2014, ch. 8, pp. 487–568.

[32] Y. Dai, S. Sun, and L. Che, "Improved dbscan-based data anomaly detection approach for battery energy storage stations," vol. 2351, 07 2022, p. 012025.

[33] H. Purohit, R. Tanabe, T. Endo, K. Suefusa, Y. Nikaido, and Y. Kawaguchi, "Deep autoencoding gmm-based unsupervised anomaly detection in acoustic signals and its hyper-parameter optimization," 2020. [Online]. Available: https://arxiv.org/abs/2009.12042