BGNN: Detection of BGP Anomalies Using Graph Neural Networks

Kevin Hoarau, Pierre Ugo Tournoux, Tahiry Razafindralambo

BGNN: Detection of BGP Anomalies Using Graph Neural Networks

Kevin Hoarau
Université de La Réunion, LIM, France
kevin.hoarau@univ-reunion.fr

Pierre Ugo Tournoux
Université de La Réunion, LIM, France
pierre.tournoux@univ-reunion.fr

Tahiry Razafindralambo
Université de La Réunion, LIM, France
tahiry.razafindralambo@univ-reunion.fr

***Abstract*—The Border Gateway Protocol (BGP) builds the communication routes at the Internet scale. Anomalies in BGP have several causes and can impact the Internet stability. BGP data traces are complex and require specific methods such as machine learning to be processed for anomaly detection. Two types of features are used to study large scale events with machine learning models: graph features or statistical features. Despite the recent interest for the concept of Graph Neural Network (GNN), there is no proposal that adapts GNN for BGP anomaly detection directly from the BGP graph. In this paper, we propose BGNN, a GNN model which detects if a node is involved in a large scale BGP anomaly. Our results show a maximum accuracy of 96% and the model can detect an anomaly after 6 minutes with 90% accuracy. These results are promising and suggest GNN for BGP anomaly detection are worth investigating.**

***Index Terms*—BGP Anomaly, Graph Neural Networks, GNN**

I. Introduction

The Border Gateway Protocol (BGP) is a routing protocol that builds the backbone of the Internet. A failure of the BGP protocol could impact any service relying on the Internet. These BGP anomalies happen for several reasons such as hardware failures, malicious attacks or mis-configurations [1]. Detection of BGP anomalies are studied using data collected from BGP projects as described in [20], [23]. This work studies BGP anomalies classified a s l arge s cale d ue t o their major impact on both BGP protocol behavior and Internet services. Large scale BGP anomalies [4] come mostly from configuration e rrors [19], m alicious w orm s pread [26], power outage [7] or hardware failure [5].

Collected BGP data traces are complex, massive and require specific m ethods s uch a s m achine l earning t o b e processed for anomaly detection. BGP data traces are analyzed and transformed into statistical features (*e.g.* counting the number of announcements, prefixes) or into graph features, *i.e.* metrics derived from the BGP graph. Machine learning models for BGP anomaly detection can be fed using graph features or statistical features extracted from BGP data traces.

The literature shows that both graph and statistical features provide excellent and similar performances on large scale BGP anomalies [15]. While graph features offer a performance increase compared to statistical features on small scale anomalies [15], graph features extraction leads to an important loss of information in comparison to the original graph representation which motivates the development of graph based end-to-end models for the detection of BGP anomalies. By leveraging all the information embedded in a graph representation of the data, this type of model could further improve the performance of BGP anomaly detection tools. Goyal et al. [12], introduce a graph embedding algorithm designed to generate stable embeddings of dynamic graphs. They showed that changes in the embedding are correlated with anomalies in the underlying context.

The recent breakthroughs in the field of Graph Neural Network (GNN) [17] have enabled the emergence of new neural networks models that can consume graph data as input. In this work, we propose a GNN based model for the detection of BGP anomalies. This model takes as input a sequence of BGP graph and predicts whether or not this sequence contains an anomaly. Therefore, this approach avoids the extraction of ad-hoc features and can leverage the graph representation of the BGP network which is most accurate and embed more information. To the best of our knowledge, this is the first work to propose GNN based model for BGP anomaly detection.

In this paper we show that, i) a GNN based model can effectively be used to detect BGP anomaly as our model achieved an accuracy score of 96%. ii) The model can generalize over a long period of time as our dataset contains events ranging from 2004 to 2021. iii) The approach can be used to detect anomalies in real-time with a response of 6 minutes for a 90% accuracy. We hope that this work will motivate the use of GNN based model for anomaly detection, and more specifically for online BGP anomaly detection. Moreover, we expect that GNN based model will also be used for small scale anomaly detection such as path hijacking or origin hijacking.

The remaining of this paper is organized as follows. In section II we provide a background on BGP, BGP anomalies and Graph Neural Networks. In section III we describe the dataset used in the paper. Section IV describes our GNN model architecture and the performance metrics used for the evaluation while section V is focused on the evaluation of the performance of our model and its analysis. We conclude the paper in section VI.

II. Background and related work

The Internet consists of Autonomous Systems (ASs) interconnected by the Border Gateway Protocol (BGP). Most

This project has received funding from the Région Réunion and the European Union - European Regional Development Fund (ERDF) as part of the INTERREG V - 2014-2020 program.

of the ASes are Internet Service Providers (identified by an ASN - AS Number) that own IP prefixes [10]. ISPs operate BGP routers that maintain TCP connections with a set of BGP neighbors to exchange routing information with other ASes. Traffic is sent through routes learned by BGP. BGP incrementally updates its set of routes. A BGP route to an IP prefix is identified by the set of ASes (namely the AS-PATH) that participate in the traffic forwarding which avoids routing loops [28].

A. BGP Anomalies

BGP anomalies are the result of failures or malfunctions of the routing protocol, protocol vulnerabilities [2], configuration errors [19], external events such as hardware failures [5] or worm spreads [26]. Some of these anomalies may lead to invalid network topologies [10] resulting in the unreachability of some prefixes. They can also cause instability or overload on the BGP routers and impact the data plan performances.

B. BGP anomaly detection using Machine Learning

The collection of BGP routing information is the cornerstone for any analysis of the BGP protocol. RouteViews [23] and RIPE RIS [20] projects have been collecting and archiving BGP data from different collectors distributed across the world since 2000. Each of these collectors receives and saves BGP updates from all its neighboring routers and updates its Routing Information Base (RIB) accordingly.

1) Statistical features: The ML models for BGP anomaly detection do not consume raw BGP data from RouteViews and RIPE RIS. They are transformed into statistical features which can be classified as i) volume features, such as the number of announcements and withdrawals, which aim to capture changes in the stability of BGP; ii) AS-PATH features, such as average AS-PATH length and the maximum edit distance, which aim to capture topological changes.

Various ML algorithms have been used to process these features *e.g.* SVM [8], [6], Naive Bayes classifiers [8], [6], decision trees [6], [18] and more recently deep learning [8], [3], [18], [4]. These works achieved good performance on the detection of large-scale anomalies such as worm spreads, massive route leaks or large-scale power outages.

2) Graph features: More recently, new works leveraged the underlying graph structure of BGP instead of statistical features [25], [12]. These dynamic graphs reflect the evolution of the BGP topology where ASes are the graph's nodes and routes are the graph's edges. In [25], the authors fed their ML model with features derived from graph theory such as centrality metrics.

In [15], the authors compared the suitability of graph and statistical features on large scale events, small scale origin and path hijacking. They found out that both graph and statistical features achieve satisfying and similar performances on large scale anomalies. However, while none of these provide satisfying performance on small scale anomalies, SVM on graph features improved the accuracy by 15%.

C. Graph Neural Networks

Conventional machine learning algorithms and neural networks are designed for tabular data and consequently cannot leverage all the information embedded in a graph representation of the data. Recently, the field of Graph Neural Networks (GNN) has emerged to overcome this issue by proposing neural networks designed to take graph representation as input. Various GNN architectures have been proposed [27] and most of them have been unified under the message passing scheme [11]. In this scheme, each node is given a value named the embedding of the node. In a GNN block, the embedding of each node is updated using the aggregation of its neighbors' embeddings. By applying multiple layers of such operation, information can be propagated over the graph's edges by one hop per layer. Moreover, in the same layer, multiple GNN blocks can be used to produce a multidimensional embedding value. Finally, the last GNN block gives the embedding value for each node of the graph.

In this work, we propose a GNN based model that takes as input a sequence of BGP graphs and predicts if this sequence contains an anomaly. A major advantage of this approach over the literature is that it avoids the extraction of ad-hoc features. It can also leverage the graph representation of the BGP network which is most accurate and embed more information than ad-hoc features. To the best of our knowledge, this paper is the first to propose a GNN based model for the detection of BGP anomaly.

III. Dataset

Our dataset includes 14 samples with 7 positive and 7 negative samples. The positive samples are extracted during the occurrences of a large scale anomaly. We arbitrarily collected the negative samples 24h before the positive sample. There is no known anomaly during the negative samples. Each of these samples is extracted from 1 hour of BGP data where the BGP graph is extracted every two minutes. It results that for each sample in our dataset we have a sequence of BGP graphs $G = G_1, ..., G_{30}$. The remaining of this section details the events included in our dataset, the data collection process and the BGP graph extraction.

A. BGP anomaly events

The anomaly events that we included in our dataset range from 2004 to 2021. First, we included 4 older events (TTNet, IndoSat, TM and AWS) for their use in previous research [25]. Second, we choose to add more recent events that reflect the modern BGP topology. The table I summaries all the events included in our dataset. For each events, we also identify the AS which is the origin of the anomaly.

B. Data collection

For both data collection and graph extraction we use BML [14]. For all the positive samples, we collect data half an hour before and after the estimated start of the event. For the negative samples, we collect data one day before each event for one hour. Therefore, for each sample, we use 1 hour

of BGP data. The data are collected from the `rrc04` and `rrc05` collectors which were chosen for their intensive use in previous research [3], [4], [25]. BGP being an incremental protocol, the BGP updates received during the hour of data collection contain only a minor fraction of the Internet's routes. This leads us to collect data during a priming period before the sample time window. Every 8 hours, Ripe RIS collectors include RIB dumps that contain all Internet prefixes reachable through the peers of the collectors. So we used a priming period of 10 hours to ensure that at least one RIB dump is collected which allows us to have a complete view of the routes available on a collector. The update messages received between the RIB dump and the observation window are used to update the routes. Thankfully, all this work is automatically carried out by BML [14].

C. Graph extraction

We denote by $G = (V, E)$ a BGP graph where V is a set of nodes corresponding to the BGP ASes and E is a set of undirected edges representing the relationship between a pair of ASes. Given a set of BGP routes, the nodes of the graph correspond to all the ASes observed in the routes. There exists an edge between two ASes (nodes) if the two ASes are adjacent in at least one of the BGP routes. Each node is weighted by the count of prefixes originated by the AS. For all the events in our dataset, we use BML to extract a snapshot of the BGP routes every 2 minutes and generate a BGP graph. The one-hour samples result in sequences of 30 graphs.

Anomaly	Date	AS Number
TTNet	Dec. 24, 2004 (9:20 UTC)	9121
IndoSat	April 2, 2014 (18:25 UTC)	4761
TM	June 12, 2015 (8:43 UTC)	4788
AWS	April 22, 2016 (17:10 UTC)	200759
Google	August 25, 2017 (3:22 UTC)	15169
ChinaTelecom	June 6, 2019 (9:57 UTC)	21217
India	April 16, 2021 (13:48 UTC)	55410

TABLE I
ANOMALY EVENTS INCLUDED IN THE DATASET

IV. GNN MODEL

The input of our model is a sequence of BGP graphs $G = G_1, ..., G_{30}$ from which it aims to output $Y = 1$ if an anomaly is detected and $Y = 0$ otherwise.

The architecture of our model is depicted in the figure 1. First, each graph G_i is given to a network composed of k GNN layers where each layer contains 8 GNN blocks. The impact of the hyperparameter k will be discussed in section V. The GNN block is a Graph Convolutional Networks [17]. For each graph G_i, an embedding of dimension 1×8 is produced for each node (8 being the number of GNN blocks). However, we only keep the embedding of the node that is the source of the anomaly (see table I). From the entire graph sequence, this extraction results in a node embedding matrix of dimension 30×8. This matrix is then flattened to obtain a vector of dimension 240. Finally, a multilayer perceptron (MLP) is used to produce the output Y based on the 240×1 vector.

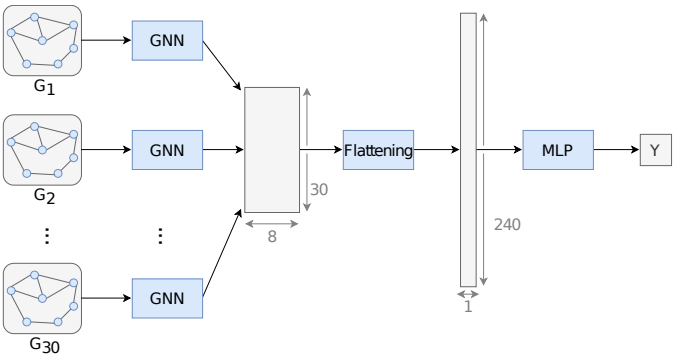

Fig. 1. Architecture of our model

For all our experiments, the model has been trained during 50 epochs with a learning rate of 0.001 using the Adam optimizer [16]. The default initialization of the GNN layers has been changed to Kaiming normal initialization [13] as we observed better convergence using this technique.

A. Evaluation metrics

To evaluate the performance of our model we rely on the following metrics:

Accuracy: The accuracy is used to evaluate the overall performance of the classifier for both positive and negative samples.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision: The precision rates the number of true positives among all the samples classified as positive.

$$Precision = \frac{TP}{TP + FP}$$

Recall: The recall rates the number of samples classified as positive among all the positive samples.

$$Recall = \frac{TP}{TP + FN}$$

F1 score: The F1 score is the harmonic mean of the precision and recall.

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

where TP (True Positive), TN (True Negative), FP (False Positive) and FN (False Negative) come from the confusion matrix [24].

As the output of a neural network is not a binary value but a value within the range $[0, 1]$ a threshold must be applied to obtain the final class. However, this threshold impacts the metrics mentioned above. For example, a low threshold will classify most of the samples as positive resulting in a high recall value but low precision. The area under the curve (AUC) which is a metric derived from the receiver operating characteristic (ROC) curve evaluates the performance of a

model at various threshold settings. The AUC represents the measure of separability of the output classes where a value of 1 indicates a perfect separability and a value of 0.5 or below indicates no separability.

V. Results

Our experiments are implemented using the *PyTorch Geometric* [9] and *scikit-learn* library. The implementations are available online[1]. We evaluate the performance of our model using a leave-one-out cross-validation scheme (LOOCV) where each event is iteratively used as a test and the others are used to train the model. When an event is used for testing both the positive and the negative sample associated to this event is used as the test set. The model then produces two outputs. When the 7 events have been used for testing, an output vector of dimension 14 is constructed by concatenating all the outputs. Finally, the output vector is used to computed the accuracy, f1 score and AUC metrics. The convergence of a neural network can vary depending on the initialization of its internal weights, it is important to evaluate the stability of the model over multiple runs. We run the model multiple times (at least 30 times with different seeds) to compute and reduce the standard deviation of the measured performance metrics.

The table II shows the performance of the model for several numbers of GNN layers varying from 1 to 16. We can see that the best performance is achieved using 4 GNN layers with an accuracy of 0.96, an f1 score of 0.96 and an AUC of 0.99. Due to the difference in the dataset/events used we do not thoroughly compare our results with the ones in the literature. However, our model based on GNN outperforms classical ML models using graph features and statistical features on large scale events. Indeed, the performance obtained in the literature is at most 0.95 accuracy and 0.95 f1 score [15], [25].

It is important to recall that the number of GNN layers corresponds to the number of hops during the message passing process. On the one hand, we can assume that with less than 4 hops, not enough data is being aggregated from the neighborhood of a node. On the other hand, values above 4 add noises to the process. With our best model, we can see that the standard deviation of the metrics is lower than 0.06 after multiple runs which shows the stability of the model.

GNN layers	Accuracy		F1 score		AUC	
	Mean	Std dev	Mean	Std dev	Mean	Std dev
1	0.89	0.05	0.88	0.05	0.86	0.03
2	0.90	0.03	0.89	0.04	0.87	0.05
4	**0.96**	**0.05**	**0.96**	**0.05**	**0.99**	**0.03**
8	0.91	0.08	0.91	0.05	0.91	0.08
16	0.78	0.11	0.72	0.20	0.79	0.09

TABLE II
Performance metrics on the test sets

[1]https://github.com/KevinHoarau/BGNN

A. GNN embedding visualization

Once the model is trained, its GNN portion can be used to produce embeddings of an input graph's nodes. For each node, this embedding is a vector in an 8-dimensional space. In this section, we want to investigate how the successive embeddings of a node are evolving with and without an anomaly on this node. A common approach for analyzing embeddings is to use a dimensionality reduction technique that projects vectors from a high-dimension space to a lower dimension space. The principal component analysis (PCA) is widely used to this end by keeping only the first n principal components (PC) from the data. Here, we keep only the first PC to obtain a 1d projection of the embeddings. The results is a time-series representing the evolution of a node's embedding.

The figure 2 shows the 1d projection of the events where blue lines correspond to the 30 embeddings of the node in the negative sample (one day before the anomaly) and the orange lines to the 30 embeddings of the node in the positive sample (during the anomaly). For all the events, we observed that the embeddings drastically change when an anomaly occurs on a node. We can also see that before the anomaly the embeddings of the positive sample are close to the embeddings of the negative sample. Thus, we assume that the embeddings do not vary significantly in a 24 hours interval. In conclusion, these results tend to show the stability of a node embedding under normal circumstance and its perturbation during an anomaly. This is a desired behavior as this signal can be leveraged by the subsequent classifier.

B. Separability exploration

The classification process is supported by the MLP portion of the model. As an input, this classifier receives a vector of dimension 240 corresponding to the flattening of the sequence of a node's embeddings. For each sample in the dataset, this vector in a 240-dimensional space is created by the GNN portion of the model. Here, we want to analyze the separation between the vectors corresponding to the negative samples and the vectors corresponding to the positive samples in this 240-dimensional space. As in the previous section, we used a PCA to project these high-dimensional vectors in a 2d space by keeping only the first 2 principal components (PC).

The figure 3 shows the 2d projection of the embedding vectors where blue points correspond to negative samples and orange points to positive samples. These vectors are obtained using the model trained using all the events except India which is used as the test set. We observed that it is possible to draw a line that separates almost all the negative samples from the positive ones. Thus, the two classes seem to be separable even in this 2d projection.

To have an intuition about the shape of the decision boundary captured by our classifier, we want to select a grid of points in the 2d space and apply our classifier to these points to know its output. By selecting a high number of points we could establish a map of the decision boundary of our model. However, our classifier cannot directly take as input a point from this 2d space so we need to project this point

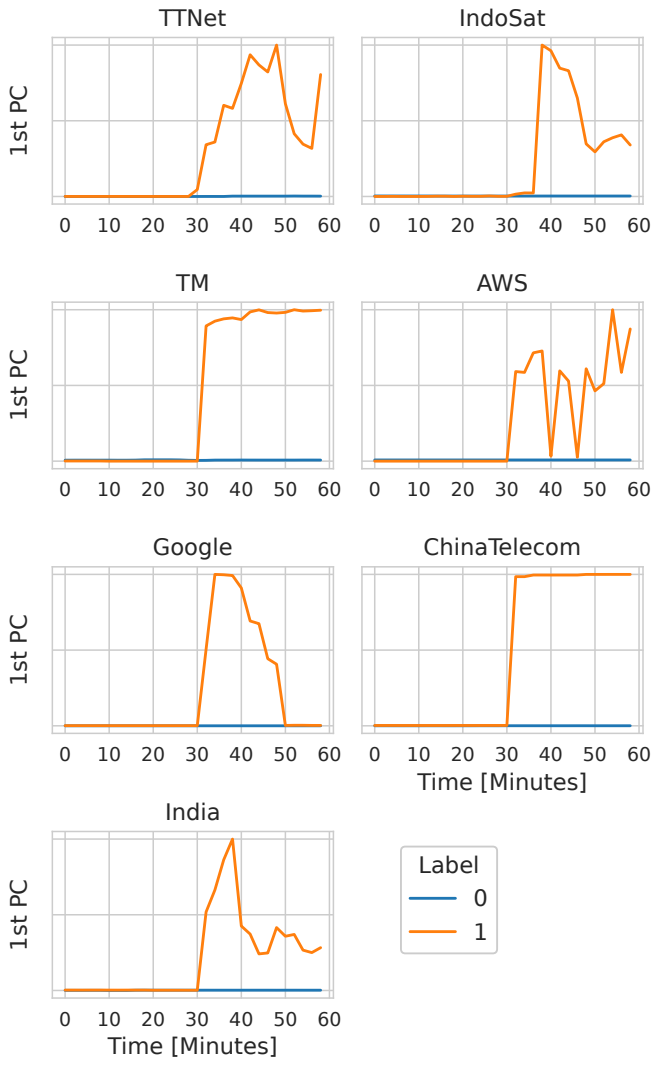

Fig. 2. AS embeddings with and without anomaly

back into the original 240-dimensional space. This approach for visualizing decision boundaries of high-dimensional space is a separate topic of research [21] and is out of the scope of this work. However, we can achieve a rough and yet imprecise mapping of the decision boundary of our model by leveraging the inverse operation of the PCA to project a 2d point back into the original space. This approach has been used to colorize the background of the figure 3. The white line in this background is the decision boundary between the two classes learned by our classifier. In this example, we can see that the classifier correctly classifies 13 samples over 14.

C. Detection time

One critical features of an anomaly detection system is its response time especially if it is to run online. However, few work in the literature address this issue by providing information about this characteristic of their model. We want to fill this gap and define the response time as the interval between the known start time of an anomaly and the time this

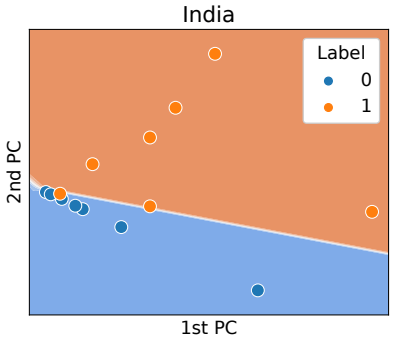

Fig. 3. Decision boundary of our classifier

anomaly can be detected. To evaluate the response time of our model, we denoted as t_0 the known start time of an anomaly and we train and test the model using data collected within the interval $[t_0 - 30, t_0 + 2]$. We gradually increase this time window by 2 minutes. Therefore, the next run is done on the interval $[t_0 - 30, t_0 + 4]$ and the last one is $[t_0 - 30, t_0 + 30]$ which correspond to our initial model presented in the previous sections. The figure 4 shows the result of this evaluation. We see from these results that 4 minutes *i.e.* a sequence of 2 graphs after the anomaly the detection accuracy, f1 score and AUC values are above 0.85 these values reach 0.90 after 6 minutes (*i.e.* a sequence of 3 graphs) and are around 0.96 after 18 minutes (*i.e.* a sequence of 9 graphs). These results are very promising for an online BGP anomaly detection and further investigations are left for future work.

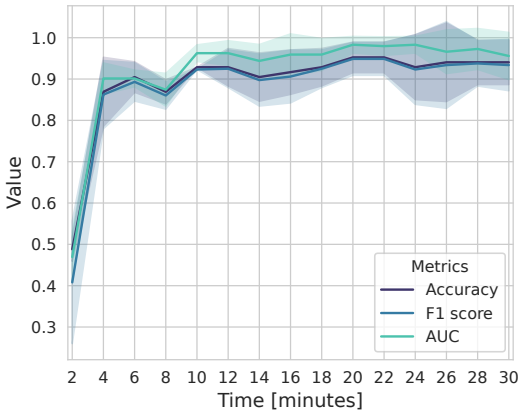

Fig. 4. Performance of the model depending on the time after the anomaly.

VI. Conclusion

Despite the sustained interest for the use of machine learning to detect BGP anomalies, the capabilities brought by Graph Neural Network (GNN) have not been leveraged yet. Our paper introduced BGNN which, to the best of our knowledge, is the first proposal and evaluation of a Graph Neural networks for anomaly detection in BGP. BGNN consists of a k-layer GNN from which we extract the embedding of a node classified as normal or anomalous by a MLP.

We first evaluated the impact of the number of layers k *i.e.* the number of hops for messages passing. It showed that the accuracy increases with k. It reaches a maximum of 96% for $k = 4$ and decreases for higher values. This suggest that local interactions carries enough information to detect an anomaly thus reducing the complexity. We then studied the first principal component of the embedding which showed a strong variation a few minutes around the expected start of the anomaly. The analysis of the embedding space using a 2d projection revealed a simple decision boundary. Finally, we evaluated how the length of the graph sequence affects BGNN's accuracy. We have shown that 6 minutes detection time is enough to achieve an accuracy of 90% and 18 minutes is enough to achieve an accuracy of 96%. Further increasing the length if the graph sequence didn't allow to further improve BGNN's accuracy.

We believe the performances achieved by such a simple GNN model open perspectives for improvement. Depending on the attributes embedded by each node of the graph (*e.g.* number of routes forwarded, number of prefix, etc.), we may be able to target different types of anomalies. More elaborated GNN such as temporal TGN [22] or a combination of GNN and LSTM might be more adapted than BGNN to integrate the temporal component of the graph sequence. GNN might also benefit to the context of smaller scale events where former ML proposals haven't been proven successful yet.

REFERENCES

[1] B. Al-Musawi, P. Branch, and G. Armitage, "BGP anomaly detection techniques: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 377–396, FebJan 2017. [Online]. Available: http://dx.doi.org/10.1109/comst.2016.2622240

[2] K. Butler, T. R. Farley, P. McDaniel, and J. Rexford, "A survey of bgp security issues and solutions," *Proceedings of the IEEE*, vol. 98, no. 1, pp. 100–122, 2009.

[3] M. Cheng, Q. Li, J. Lv, W. Liu, and J. Wang, "Multi-scale lstm model for bgp anomaly classification," *IEEE Transactions on Services Computing*, 2018.

[4] M. Cosovic, S. Obradovic, and E. Junuz, "Deep learning for detection of bgp anomalies," in *International Work-Conference on Time Series Analysis*. Springer, 2017, pp. 95–113.

[5] J. H. Cowie, A. T. Ogielski, B. Premore, E. A. Smith, and T. Underwood, "Impact of the 2003 blackouts on internet communications," *Preliminary Report, Renesys Corporation (updated March 1, 2004)*, 2003.

[6] I. O. de Urbina Cazenave, E. Köşlük, and M. C. Ganiz, "An anomaly detection framework for bgp," in *2011 International Symposium on Innovations in Intelligent Systems and Applications*, June 2011, pp. 107–111.

[7] S. Deshpande, T. Ho, M. Thottan, and B. Sikdar, "An online mechanism for bgp instability detection and analysis," *IEEE Transactions on Computers*, vol. 58, no. 11, pp. 1470–1484, nov 2009.

[8] Q. Ding, Z. Li, P. Batta, and L. Trajković, "Detecting bgp anomalies using machine learning techniques," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2016, pp. 003 352–003 355.

[9] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[10] L. Gao, "On inferring autonomous system relationships in the internet," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 733–745, Dec 2001.

[11] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1263–1272.

[12] P. Goyal, N. Kamra, X. He, and Y. Liu, "Dyngem: Deep embedding method for dynamic graphs," *arXiv preprint arXiv:1805.11273*, 2018.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[14] K. Hoarau, P.-U. Tournoux, and T. Razafindralambo, "BML: an efficient and versatile tool for BGP dataset collection," in *WS22 IEEE ICC 2021 the 3rd International Workshop on Data Driven Intelligence for Networks and Systems (WS22 ICC'21 Workshop - DDINS)*, Montreal, Canada, Jun. 2021.

[15] ——, "Suitability of graph representation for bgp anomaly detection," in *2021 IEEE 46th Conference on Local Computer Networks (LCN) (LCN 2021)*, Edmonton, Canada, Oct. 2021.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[18] Y. Li, H. J. Xing, Q. Hua, X. Z. Wang, P. Batta, S. Haeri, and L. Trajković, "Classification of bgp anomalies using decision trees and fuzzy rough sets," in *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2014, pp. 1312–1317.

[19] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding bgp misconfiguration," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 3–16, 2002.

[20] RIPE, "Routing information service (RIS)." [Online]. Available: https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/routing-information-service-ris

[21] F. Rodrigues, M. Espadoto, R. Hirata, and A. C. Telea, "Constructing and visualizing high-quality classifier decision boundary maps," *Information*, vol. 10, no. 9, p. 280, 2019.

[22] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," *arXiv preprint arXiv:2006.10637*, 2020.

[23] RouteViews, "Routeviews - university of oregon route views project." [Online]. Available: http://www.routeviews.org/routeviews/

[24] C. Sammut and G. I. Webb, *Encyclopedia of machine learning*. Springer Science & Business Media, 2011.

[25] O. R. Sanchez, S. Ferlin, C. Pelsser, and R. Bush, "Comparing machine learning algorithms for bgp anomaly detection using graph features," in *Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks*, 2019, pp. 35–41.

[26] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S. F. Wu, and L. Zhang, "Observation and analysis of bgp behavior under stress," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 2002, pp. 183–195.

[27] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *arXiv preprint arXiv:1901.00596*, 2019.

[28] S. H. Y. Rekhter, T. Li, "A border gateway protocol 4 (bgp-4)," Network Working Group, IETF, RFC 4271, 2006. [Online]. Available: https://tools.ietf.org/html/rfc4271