UNSUPERVISED REPRESENTATION LEARNING FOR BGP ANOMALY DETECTION USING GRAPH AUTO-ENCODERS

Kevin Hoarau[1], Pierre Ugo Tournoux[1], Tahiry Razafindralambo[1]
[1]LIM, University of Reunion Island

NOTE: Corresponding author: Kevin Hoarau, kevin.hoarau@univ-reunion.fr

Abstract – *The Border Gateway Protocol (BGP) is crucial for the communication routes of the Internet. Anomalies in BGP can pose a threat to the stability of the Internet. These anomalies, caused by a variety of factors, can be challenging to detect due to the massive and complex nature of BGP data traces. Various machine learning techniques have been employed to overcome this issue. The traditional approach involves the extraction of ad hoc features, which, although effective, results in a significant loss of information and may be biased towards a certain type of anomaly. A recent supervised machine learning pipeline learns representations from BGP graphs derived from BGP data traces. Although this solution achieves good anomaly detection results, the representations learned are specific to the types of anomalies within the training data. To overcome this limitation, in this paper, we propose to learn the representations of normal BGP behaviour in an unsupervised manner using a Graph Auto-Encoder (GAE). This approach ensures that the representations are not limited to the specific set of anomalies included in the training set. These representations associated with a Multi-Layer Perceptron (MLP)-based detector allowed to achieve an accuracy rate of 99% in detecting large-scale events, outperforming previous literature results.*

Keywords – BGP anomaly, graph auto-encoders, graph neural network (GNN)

1. INTRODUCTION

The Border Gateway Protocol (BGP) is a critical routing protocol that forms the foundation of the Internet. Anomalous behaviour of this protocol has the potential to impact all services relying on the Internet. BGP anomalies can have several causes such as hardware failure, malicious attacks, or misconfigurations [1]. The work presented in this paper specifically focuses on BGP anomalies classified as large-scale due to their major impact on both BGP protocol behaviour and Internet services. Large-scale BGP anomalies are mainly caused by configuration errors [2], malicious worm spread [3], power outage [4] or hardware failure [5].

As highlighted by the pipeline described in Fig. 1, the study of BGP anomalies requires data traces collected from BGP data collection projects or archives, such as RIS [6] or RouteViews [7]. Such traces are complex and require a significant amount of preprocessing prior to being fed to machine learning algorithms. As seen in Fig. 1, part of the preprocessing may involve transforming them into either statistical features (*i.e.* count of announcements, prefixes [8, 9, 10, 11]) or graph features (*e.g.* eccentricity, centrality [12, 11]). Machine learning models used for BGP anomaly detection are fed either with graph features or statistical features extracted from the BGP data traces (*cf.* anomaly detection block of Fig. 1).

The literature highlights that graph and statistical features have demonstrated similar performance for detecting large-scale BGP anomalies [11]. While features extracted from the graph may allow extracting complementary and relevant patterns, the extraction of graph features still induces a significant loss of information compared to the original graph. This observation motivates the development of end-to-end graph-based models for BGP anomaly detection. By allowing the Machine Learning (ML) model to leverage all the information embedded within the BGP graph, this approach has the potential to significantly enhance the performance of BGP anomaly detection tools.

This has been enabled by the recent breakthroughs in the field of Graph Neural Network (GNN) [14] with the emergence of new neural network models that can handle graphs as input. In the work from Hoarau *et al.* [13], a GNN-based model was employed to detect BGP anomalies. As depicted in Fig. 2, the model from [13] takes a sequence of BGP graphs as input and predicts the presence of anomalies. Training the model requires well-labelled sequences of graphs collected during normal BGP behaviour and during BGP anomalies. The approach circumvents the extraction of ad hoc features and instead learns the representation during model training (*cf.* Fig. 1). While it achieves a 96% accuracy in large-scale anomaly detection, limitations can be identified. The representations were specifically learnt for a particular type of anomalies (included in the training set) and may not generalize to other types or future anomalies that may significantly differ. Additionally, for a given sample of the dataset only one node is used to compute the prediction error, which penalized the diversity of captured patterns. Finally, the training requires carefully labelled data on past BGP anomalies, which is scarce.

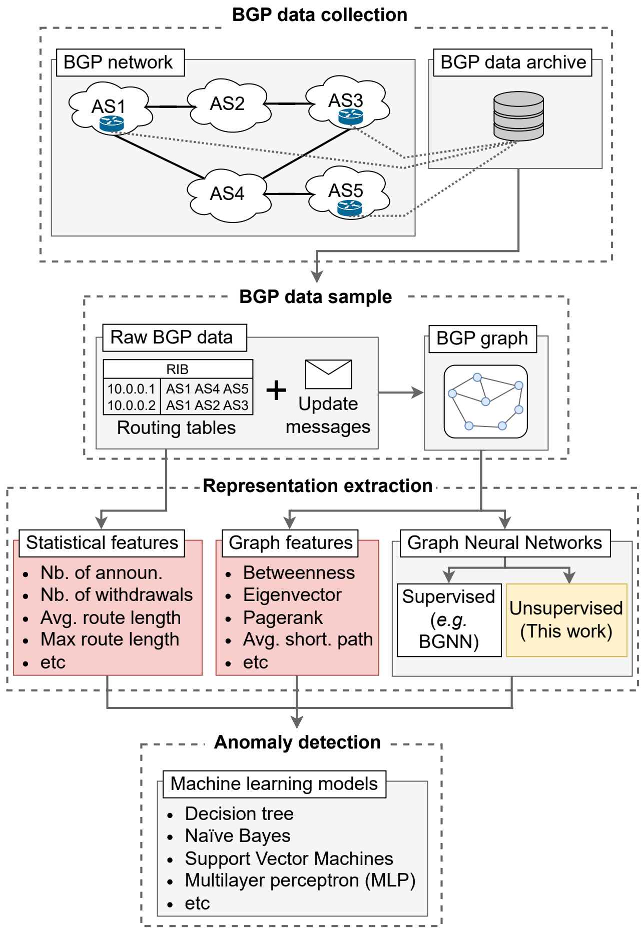

Fig. 1 – Overview of machine learning-based pipelines for the detection of BGP anomalies

In this paper, and unlike in previous work [13], we propose the use of a Graph Auto-Encoder [15] (GAE) to learn BGP graph representations in an unsupervised manner. This approach offers multiple benefits.

- The representations are learned from BGP graphs without anomalies, leading to greater generalization potential.
- For each sample of the dataset, every node of the BGP graph (around $60K$) instead of one are used to compute the error of the model, allowing to capture more diverse pattern.
- The approach does not require labelled data, allowing for the utilization of abundant unlabelled BGP data [6, 7].

Our proposal uses an MLP to classify the representations, which achieved an accuracy rate of 99% in detecting large-scale BGP anomaly events, outperforming previous literature results from [12, 13]. This paper confirms that the combination of unsupervised learning and graph representation for BGP improves anomaly detection. This may open new research directions for smaller scale anomaly detection which are yet to be detected by an ML scheme.

The remainder of this paper is organized as follows. In Section 2 we provide a background on BGP, BGP anomalies and GNN. In Section 3 we

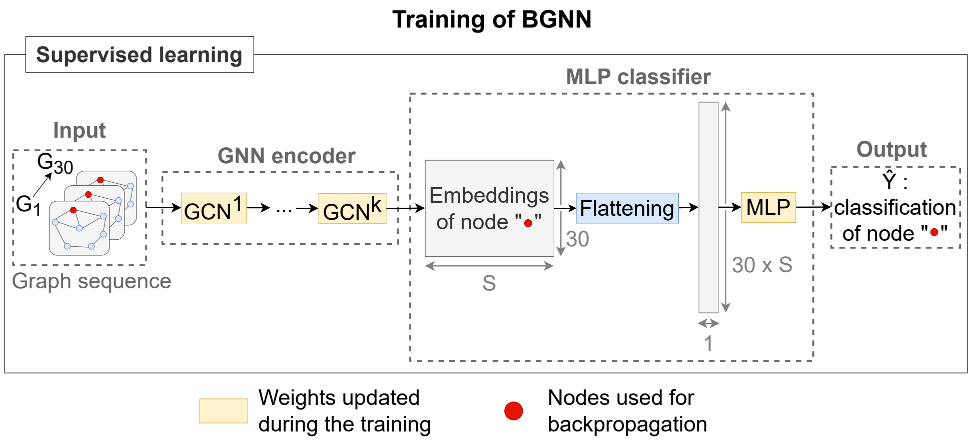

Fig. 2 – Training of the supervised BGNN model [13]

describe the dataset used in the paper. Section 4 describes our GAE model architecture and presents how the training was performed. Section 5 focuses on the detection of BGP anomalies using the representations learned by the GAE. Section 6 discusses the results of this paper and presents some direction for future work in the field of the detection of BGP anomalies. We conclude the paper in Section 7.

2. BACKGROUND AND RELATED WORK

The Internet is composed of Autonomous Systems (ASes) interconnected by BGP. The majority of these ASes are Internet service providers that own IP prefixes [16] and are identified by an AS Number (ASN).
The route to an IP prefix in BGP is identified by the series of ASes (namely the AS-PATH) that participate in the traffic forwarding, which avoids routing loops [17].

2.1 BGP anomalies

BGP anomalies can be caused by a variety of factors such as failures or malfunctions of the routing protocol, protocol vulnerabilities [18], configuration errors [2], external events such as hardware failure [5] or worm spread [3]. These anomalies can cause instability or overload on the BGP routers and impact the data plan performances. Worse, they can also lead to invalid network topologies [16] resulting in the unreachability of some prefixes.

2.2 BGP anomaly detection using features-based models

BGP routing data is essential for any analysis of the BGP protocol. Thankfully, projects such as RouteViews [7] and RIPE RIS [6] have been collecting and archiving BGP data from different collectors distributed across the world since 2000 (see block BGP data collection of Fig. 1). These collectors are connected to several neighbouring BGP routers, from which they receive and save BGP update messages. These messages are also used to update their Routing Information Base (RIB) that they also save periodically (see block BGP data sample of Fig. 1). From this raw BGP data, a sequence of BGP graphs that reflect the evolution of the BGP network can be extracted, which is automatically carried out by a recent tool presented in [19].

2.2.1 Statistical features

Statistical features (see Fig. 1) extracted from raw BGP data can be classified as: i) volume features, such as the number of announcements and withdrawals; ii) AS-PATH features, such as average AS-PATH length, and the maximum edit distance. Various work, using statistical features, achieved good performance on the detection of large-scale anomalies using conventional ML models such as SVM [8, 9], Naive Bayes classifiers [8, 9], decision trees [9] and deep learning [8, 10].

2.2.2 *Graph features*

In [12], the authors used BGP graphs to derive features from graph theory as input to their ML models (*cf. Fig.* 1 (Graph features))
In [11], the authors show that both graph and statistical features achieve satisfying and similar performances on large-scale anomalies. However, none of the statistical and graph-based features achieves satisfying performance on a small-scale origin and path hijacking. Nevertheless, accuracy of SVM with graph features outperforms statistical features by 30% [11].

2.3 Representation learning from BGP graphs

Machine learning algorithms and vanilla neural networks are not designed to handle graphs as input. Recently, the field of GNN [20] has emerged to overcome this issue by proposing neural networks that can directly consume graphs and learn representations that can be fed to conventional neural network architectures (see Fig . 1 (Graph Neural Networks)).
In a previous work [13], a GNN-based model takes as input a sequence of BGP graphs and predicts if a given node is the victim of an anomaly within this sequence. The architecture of this model is depicted in Fig . 2. Despite the good performance of this model, the representations are specifically learned for the type of anomalies included in the dataset and could fail to generalize to other types of anomalies. Additionally, only one node is used to compute the prediction error of the model during the training, which limits the diversity of learned patterns. Furthermore, this approach requires carefully labelled data about past BGP anomalies that are not massively available.
In this work, we suggest using a GAE [15] to learn representations in an unsupervised manner. The architecture of the proposed model is depicted in Fig . 3 (phase 1). This approach has three key benefits: (1) the representations are learned independently of anomalies, leading to greater generalization potential, (2) all the nodes of the graph are used to compute the error of the model during the training, allowing for the capture of more diverse patterns, and (3) the model training does not require labelled data, enabling the use of large amounts of unlabelled data.

3. DATASET

The dataset used in this work consists of 14 samples, comprised of 7 positive samples and 7 negative samples. The positive samples are extracted during the occurrence of large-scale anomalies, while the negative samples are arbitrarily collected 24 hours prior to the corresponding positive sample. Manual inspection didn't show any sign of large-scale anomaly during negative samples. Each of these samples is extracted from 1 hour of BGP data, where the BGP graph is extracted every two minutes. This results in a sequence of 30 BGP graphs for each sample in the dataset $G = G_1, \ldots, G_{30}$. This section provides further details on the events included in the dataset, the data collection process and the BGP graph extraction.

3.1 BGP anomaly events

The anomaly events used in our dataset span from 2004 to 2021 and include both older and more recent events. Four older events (TTNet, IndoSat, TM, and AWS) were included due to their frequent use in previous research [12]. Additionally, more recent events were selected to better reflect the current BGP topology. A summary of all events in the dataset is provided in Table 1. For each event, we also identify the AS which is the origin of the anomaly.

3.2 Data collection

Both data collection and graph extraction are performed using the BML tool [19]. For positive samples, BGP data is collected half an hour before and after the estimated start of the event, resulting in one hour of data per sample. For negative samples, BGP data is collected one day before each event for one hour. The data is

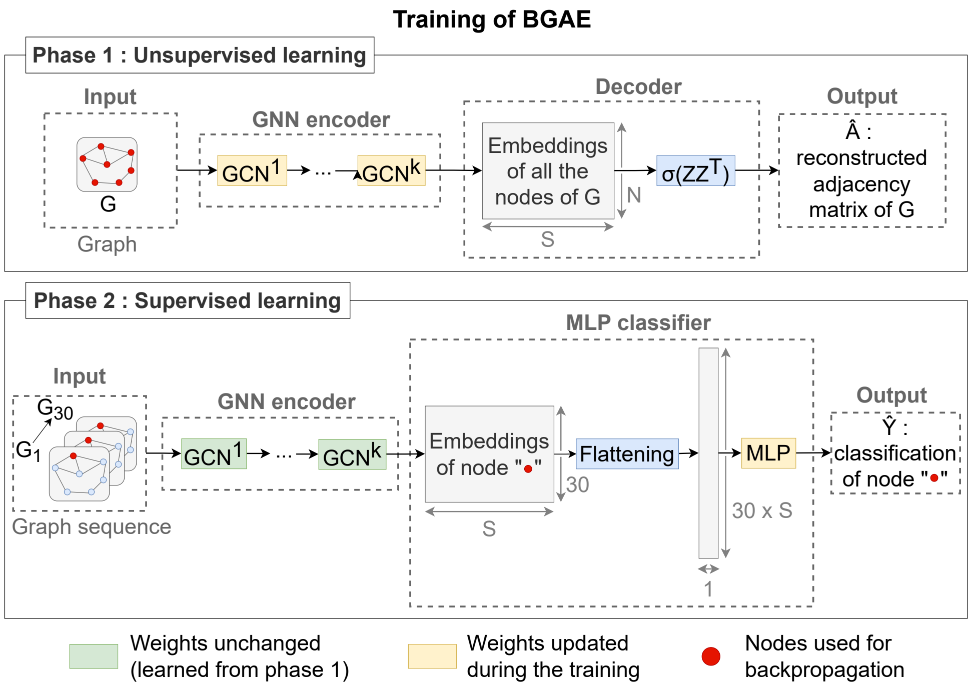
Training of BGAE
Phase 1 : Unsupervised learning
Input
G
Graph
GNN encoder
GCN1 → ... → GCNk
Decoder
Embeddings of all the nodes of G
N
S
σ(ZZT)
Output
Â : reconstructed adjacency matrix of G
Phase 2 : Supervised learning
Input
G30
G1
Graph sequence
GNN encoder
GCN1 → ... → GCNk
MLP classifier
Embeddings of node "•"
30
S
Flattening
30 x S
1
MLP
Output
Ŷ : classification of node "•"
Weights unchanged (learned from phase 1)
Weights updated during the training
Nodes used for backpropagation

Fig. 3 – Training of the BGAE model (this work)

collected from the `rrc04` and `rrc05` collectors, which were chosen for their intensive use in previous research [10, 12]. Due to the incremental nature of BGP, the BGP updates received during the hour of data collection only represent a small portion of the Internet's routes. This leads us to collect data during a priming period before the sample time window. Every eight 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 these tasks are automatically handled by BML [19].

3.3 Graph extraction

We denote by $G = (V, E, X)$ a BGP graph where V is a set of nodes corresponding to the BGP ASes, E is a set of undirected edges representing the relationship between a pair of ASes, and $X \in \mathbb{R}^{N\times 1}$ is the nodes weight matrix where N is the number of nodes of the graph. 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 n has a weight X_n that corresponds to the number 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 two minutes and generate a BGP graph. The one-hour samples result in sequences of 30 graphs.

Table 1 – Anomaly events included in the dataset

Anomaly	Date	AS #
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
ChinaT.	June 6, 2019 (9:57 UTC)	21217
India	April 16, 2021 (13:48 UTC)	55410

4. UNSUPERVISED REPRESENTATION LEARNING

4.1 The Graph auto-encoder model

The architecture of the GAE [15] model is depicted in Fig. 3 (phase 1). The model is composed of two main parts: the encoder and the decoder. The objective of the encoder is given an input graph to produce a latent representation Z of this graph. This latent representation is a matrix of dimension $N \times S$ where N is the number of nodes in the input graph and S is the dimension of the latent space. A vector $Z_n \in R^S$ with $n \in N$ called the embedding of the node n is the representation of this node in the latent space. Given, the latent representation Z, the objective of the decoder is to reconstruct the adjacency matrix A of the graph. Finally, given a reconstructed adjacency matrix $\hat{A}$ the quality of the reconstruction can be evaluated using the reconstruction error:

$$\begin{aligned} RE = &\frac{1}{Pe}\sum_{i=1}^{N}\sum_{j=1}^{N} -(A_{ij} * \log(\hat{A}_{ij})) \\ &+ \frac{1}{Ne}\sum_{i=1}^{N}\sum_{j=1}^{N} -((1-A_{ij}) * \log(1-\hat{A}_{ij})) \end{aligned} \tag{1}$$

where $Pe = \sum_{i=1}^{N}\sum_{j=1}^{N} A_{ij}$ is the number of positive edges and $Ne = \sum_{i=1}^{N}\sum_{j=1}^{N}(1-A_{ij})$ is the number of negative edges. An edge between two nodes i and j is positive if the nodes are connected ($A_ij = 1$) and negative if they are not ($A_ij = 0$).

4.1.1 Encoder

The encoder part of the model is a k layer GNN based on the Graph Convolutional Network (GCN) [14] operator. At a layer l of the GNN, a latent representation $Z^{(l)}$ is produced as:

$$Z^{(l)} = GCN^{(l)}(Z^{(l-1)}, A) = ReLU(\tilde{A}Z^{(l-1)}\theta^{(l)}) \tag{2}$$

where $\tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix, D is the degree matrix, $\theta^{(l)}$ is the weight matrix for the layer l. For the first layer, $Z^{(0)} = X$ and $\theta^{(1)}$ is a matrix of dimension $1 \times S$. For the layer $l > 1$, $\theta^{(l)}$ is a matrix a of dimension $S \times S$. The final latent representation of the graph is $Z = Z^{(k)}$, the output of the last GCN layer.

4.1.2 Decoder

Given a latent representation Z of the graph, the reconstructed adjacency matrix $\hat{A}$ is computed as the inner product between the node embeddings:

$$\hat{A} = \sigma(ZZ^T) \tag{3}$$

where σ is the sigmoid activation function.

4.2 Model training

Our experiments are implemented using the *PyTorch Geometric* [21] and *scikit-learn* libraries and are available online[1].

The aim of the training phase is to adjust the model's parameters, $\theta^{(0)}, \ldots, \theta^{(k)}$, in order to capture meaningful latent representations. To this end, the model is trained in an unsupervised manner on BGP graphs that do not contain anomalies (only negative samples). In order to assess the model's generalization capabilities on unseen data, a portion of the graphs must be reserved for testing. As we want to test the model on every event present in our dataset, we train our model using a Leave-One-Out-Cross-Validation (LOOCV) scheme where each event is iteratively used as the test set and the others are used to train the model. Therefore, seven models are trained, one for each event in the dataset. During the training phase, only the negative samples are used and the Adam optimizer with a learning rate of 0.001 is used to minimize the reconstruction error. With each negative sample comprised of 30 graphs, the total number of graphs used for training is $6 * 30 = 180$. To reduce computational cost, the negative edges are sampled for the calculation of the reconstruction error. The number of negative edges sampled for each graph is set to match the number of positive edges, and the sampling is performed once for all experiments.

4.3 Early stopping

One of the difficulties of training a machine learning model for representation learning is to avoid overfitting, which can produce represen-

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

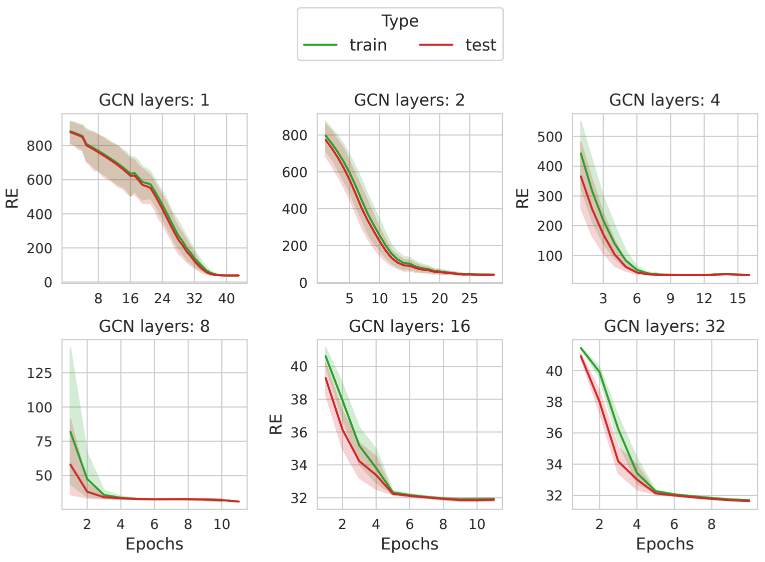

Fig. 4 – Reconstruction error during the training of the GAE

tations that are too specific for the samples in the dataset and that do not generalize to unseen data. In our experiment, our model is trained on 100 epochs and with a batch size of 16 but we introduce an early stopping [22] mechanism that can stop the training of the model when the reconstruction error reaches a plateau.
First, we measured RE_0, the initial reconstruction error when no training has been made. Then at each epoch t we compute the difference between the actual reconstruction error and the previous one:

$$\Delta_{RE} = RE_{t-1} - RE_t \tag{4}$$

During the training, if Δ_{RE} is lower than a convergence threshold λ then stagnation is detected. As short-lived stagnation may happen, we only stop the training when the stagnation is maintained for five epochs. The convergence threshold is defined as a fraction of the initial reconstruction error:

$$\lambda = RE_0 \times \epsilon \tag{5}$$

Fig. 4 shows the evolution of reconstruction error during the training of the model with a value of $\epsilon = 0.005$. We can see a few epochs are needed to reach the plateau and that the early stopping mechanism is doing well at detecting the stagnation.

4.4 Visualization of reconstruction error

We can make the assumption that when a node n is a victim of an anomaly, its embedding vector Z_n will also be impacted. And as our model was not trained on BGP graphs containing anomalies, we hypothesize it will fail to reconstruct the adjacency of this node. For a sequence of graphs and a given node n, we propose to visualize the reconstruction of the adjacency of n. To this end, for each graph of the sequence, the reconstruction error is computed using all the positive and negative incident edges of n.

Fig. 5 shows the evolution of the reconstruction error of a node with and without an anomaly. For each event, the model trained on the remaining events is applied to both the negative and positive sequence of graphs and subsequently, the reconstruction error is computed for the adjacency of the node victim of the anomaly. Two observations can be made from Fig. 5. First, without an anomaly (i.e. negative sequences), the reconstruction error is stable. Second, the reconstruction error significantly increases during an anomaly. The combination of these two properties is practical for the detection of anomalies.

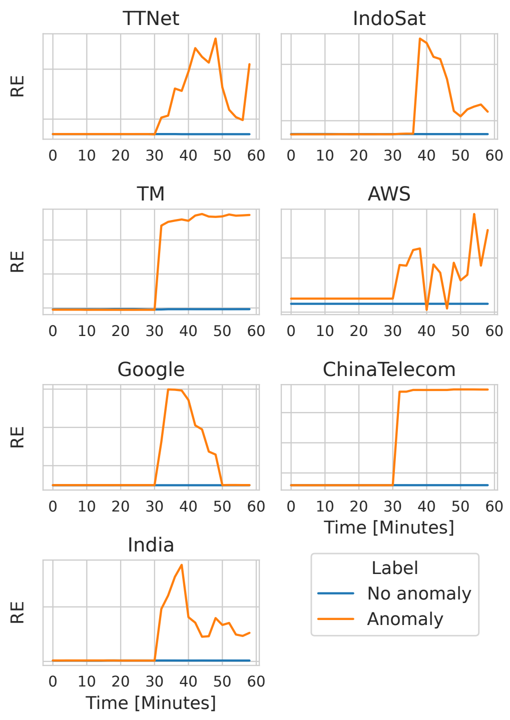

Fig. 5 – Node adjacency Reconstruction Error (RE) with and without an anomaly

5. ANOMALY DETECTION

In this section, we investigate how the representation learned using the GAE model can be used to detect BGP anomalies. The problem considered is a time-series classification problem. From a sequence of 30 graphs $G^1, \dots, G^{30}$, we extract a time-series $Z = z_n^1, \dots z_n^{30}$ where $z_n^t \in \mathbb{R}^S$ is the latent representation of the node n for the graph G^t. For a given time-series Z, the objective is to predict if the sequence contains an anomaly.

For this task, we used an MLP as this model has achieved good performances in the BGNN model of [13]. This model cannot consume time-series hence the input should first be flattened. Therefore, the time-series $Z = z_n^1, \dots z_n^{30}$ is transformed into a vector $Z' \in \mathbb{R}^{30S}$. The vector Z' is then fed to the MLP that produces a binary output $\hat{y}$ corresponding to the presence or the absence of an anomaly.

The pipeline for the detection of an anomaly is depicted in Fig. 3 (phase 2). In comparison to previous work by Hoarau *et al.* (BGNN), the time-series Z is produced by the pre-trained encoder of the GAE (step 1) whereas in BGNN a GNN is trained in an end-to-end fashion (*cf.* Fig. 2).

5.1 Training of the MLP

We used an LOOCV to evaluate the performances of the models. For each test event, the models produce two outputs, one for the negative sequences and one for the positive sequences of graphs. When the seven 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 compute the accuracy, F1 score and AUC metrics [23]. 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 trained the entire pipeline (*cf.* Fig. 3) multiple times (30 times with different seeds) to compute and reduce the standard deviation of the measured performance metrics.

5.2 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 [23].

Due to the fact that a neural network produces a value within the range of $[0, 1]$ rather than a binary value, a threshold needs to be applied in order to determine the final class. However, the choice of threshold has an impact on the metrics used to evaluate the model. For instance, if a low threshold is used, most of the samples will be classified as positive, leading to a high recall value but with low precision. To assess the performance of the model at different threshold settings, the Area Under the Curve (AUC) is computed from the Receiver Operating Characteristic (ROC) curve (see Fig. 6). The AUC reflects the degree of separability of the output classes, where a score of 1 indicates perfect separability and a score of 0.5 or below indicates no separability.

5.3 Exploration of the GAE hyperparameter space

Two hyperparameters of the GAE model have a significant impact on the representation capability of the model and subsequently on the performance of anomaly detection. First, the number of GNN layers used in the encoder that we denote k impacts how far information can propagate over the edges of the graph to compute the latent representations of the nodes [24]. In other words, a node's representation is influenced only by nodes within a distance of k or less. In theory, large values of k would allow the detection of anomalies from distant nodes. However, deep GNN models (i.e., those with a high k) are prone to over-smoothing and vanishing gradient problems [25], which can compromise the quality of the learned representations. Therefore, a balance should be found to still capture information from distant nodes without introducing these types of issues.

The second hyperparameter to consider is S, which represents the dimension of the latent representation of the nodes. It corresponds to the number of GNN blocks per layer and determines the expressiveness of the representations learned by the model. Low values of S will result in representations that fail to capture intricate patterns in the data, while excessive values may capture irrelevant information.

Fig. 7 illustrates the impact of these parameters on the accuracy of anomaly detection. For each combination of these parameters, we trained the GAE and applied the MLP to the representations obtained to measure the detection accuracy. The results show that the optimal combination of parameters, which yields 0.99 accuracy, is $k = 4$ and $S = 8$. A clear separation between the upper left and bottom right portions of the figure suggests that better performance is achieved when $S \geq k$. This finding indicates that deeper GNNs require larger representation sizes.

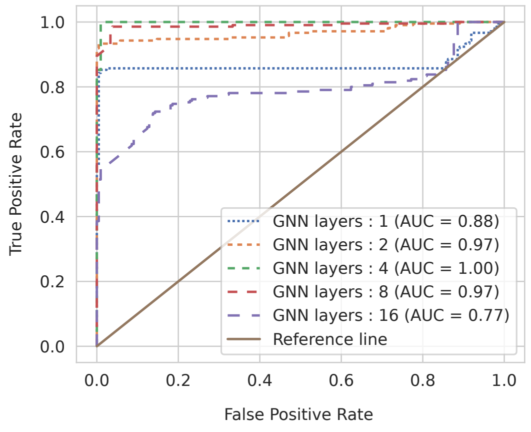

Fig. 6 – Receiver Operating Characteristic (ROC) curve

5.4 Results

Table 2 shows the performance of the MLP using representation learned by the GAE trained with hyperparameters $k = 4$ (i.e the number of GNN layers) and $S = 8$ (i.e. the number of GNN blocks per layer). These values are selected as they lead to the best accuracy score in the previous section.

We can see that the overall performances of the model are high with an accuracy and an F1 score of 0.99 and the standard deviation of 0.03 indicates the stability of the model over multiple random initializations. Furthermore, the AUC of

Table 2 – Performance metrics on the test sets

Test set	**Accuracy**		**F1 score**		**AUC**	
	Mean	Std dev	Mean	Std dev	Mean	Std dev
TTNet	1.00	0.00	1.00	0.00	1.00	0.00
IndoSat	1.00	0.00	1.00	0.00	1.00	0.00
TM	1.00	0.00	1.00	0.00	1.00	0.00
AWS	1.00	0.00	1.00	0.00	1.00	0.00
Google	1.00	0.00	1.00	0.00	1.00	0.00
ChinaTelecom	0.93	0.17	0.87	0.34	1.00	0.00
India	0.97	0.12	0.98	0.08	1.00	0.00
Overall	**0.99**	**0.03**	**0.99**	**0.03**	**1.00**	**0.01**

1 shows that the positive and negative samples could be perfectly classified by the model using a well-tuned decision threshold. Fig. 6 shows the ROC curves and AUC values depending on the number of layers k of the GAE encoder. It consolidates the choice of, $k = 4$ as stated in the previous section.

It is worth noting, that only the more recent anomalies (ChinaTelecom 2019, India 2021) are sometimes missclassified by the model (*cf.* Table 2). One explanation could be that the GAE is mostly trained on older graphs, and therefore, the model is not well adapted to extract meaningful representations for these newer graphs. In future work, this issue could be overcome by training a GAE using graphs extracted during the same year of the anomaly. This would not be possible with supervised graph representation learning such as BGNN [13]. To dig deeper into the improvement brought by the GAE, Table 3 presents a comparison of our proposal (BGAE) with a previous work by Hoarau *et al.* (BGNN) for different values of k, the number of GNN layers. Two observations can be made. First, both BGAE and BGNN achieve higher performances for $k = 4$. Second, except for $k = 16$, BGAE always achieves a higher performance than BGNN for the same value of k.

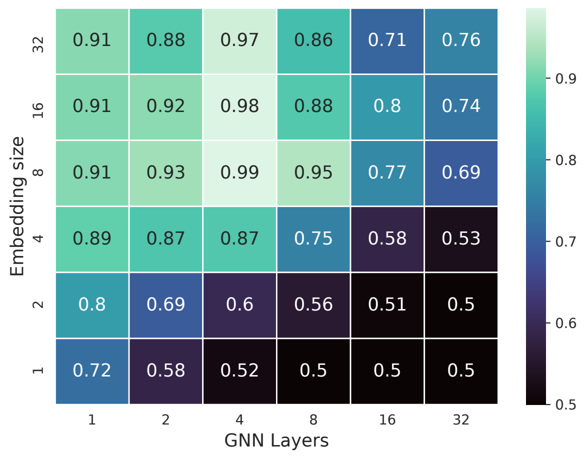

Fig. 7 – Accuracy of the MLP depending on the GAE hyperparameters

6. DISCUSSION AND PERSPECTIVES

The impact of several parameters is yet to be evaluated, *i.e.* the number of BGP collectors, the distribution of these collectors in the Internet topology, as well as their position relative to the attacker or the victim might be worth investigating. As mentioned in Section 5, using graph representation learned from a past Internet era to detect the modern era probably degrades the performance. The impact of the time difference between the training period and the anomalies should be evaluated. We have the intuition that the GAE should be trained with the most recent data available. However, in this work for comparison purposes, we preferred to comply with the ML pipeline from the authors of a previous proposal in which their training set mixes data from events up to fifteen years apart.

Table 3 – Comparison of the performances of BGNN [13] and BGAE depending on the number of GNN layers.

GNN layers	Model	Accuracy		F1 score		AUC	
		Mean	Std dev	Mean	Std dev	Mean	Std dev
1	BGNN	0.89	0.05	0.88	0.05	0.86	0.03
	BGAE	0.91	0.04	0.91	0.03	0.88	0.00
2	BGNN	0.90	0.03	0.89	0.04	0.87	0.05
	BGAE	0.93	0.04	0.92	0.05	0.97	0.04
4	BGNN	0.96	0.05	0.96	0.05	0.99	0.03
	BGAE	**0.99**	**0.03**	**0.99**	**0.03**	**1.00**	**0.01**
8	BGNN	0.91	0.08	0.91	0.05	0.91	0.08
	BGAE	0.95	0.10	0.92	0.18	0.97	0.09
16	BGNN	0.78	0.11	0.72	0.20	0.79	0.09
	BGAE	0.77	0.11	0.72	0.17	0.77	0.10

As discussed in Section 4, GAE will allow gathering a much richer set of training data to produce the graph representation. The optimal duration of the training data should be evaluated.

In BGAE's unsupervised representation learning, all nodes of the BGP graph are used for computing the model's error. This differs from the seminal BGNN model which uses only a single node. While this characteristic allows BGAE to capture more diverse representations of BGP nodes it may raise questions regarding the cost of computing such a topology-wide reconstruction error. During the GAE training, this only adds a marginal computation cost compared to the most intensive tasks which are the forward and backward passes of the GNN architecture. Moreover, when BGAE is deployed for anomaly detection, it is only the model inference that has to be performed in a timely manner. During this phase, the reconstruction error is not computed, making BGNN and BGAE models equivalent in terms of complexity.

The 0.99 accuracy achieved through the use of a GAE [15] could lead the reader to erroneously consider that there is no significant margin left for the improvement of BGP anomaly detection. However, while most of the research effort has focused on large-scale route leaks, BGP anomalies also comprise origin and path hijacking which are usually much smaller events. Fig. 8 describes the BGP anomaly detection accuracy achieved by various machine learning models depending on the type of events. It shows that feature-based models' [11, 26] accuracy is somewhat satisfying on route leaks (.88 accuracy) but they may not be of any practical use to date for origin hijacking (.57 accuracy) and path hijacking (.67 accuracy). To the best of our knowledge, Graph-based models have not yet been adapted and evaluated for origin and path hijacking anomaly detection. The significant enhancement they have made to large-scale anomaly detection suggests that it might be a worthwhile endeavour.

To date, none of the graph-based models that have been used on BGP data intrinsically integrate the temporal dimension of the BGP data. The temporal GNN model, as outlined in [27], holds great potential for this undertaking.

7. CONCLUSION

The detection of BGP anomalies through machine learning has long been hindered by the limitations of the input features: either statistical features (such as count of announcements, prefixes) or graph features (such as eccentricity, centrality) used by the models. The extraction or computation of such features from raw BGP data resulted in a significant loss of information, making it difficult to detect BGP anomalies.

Recent work [13] has shown that the use of GNN-based models overcome this limitation by

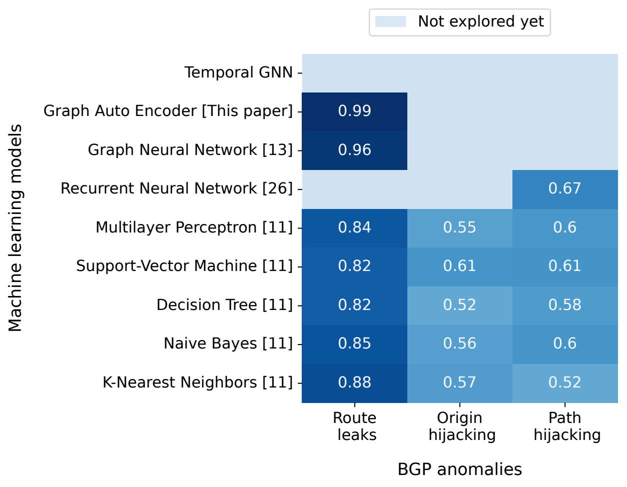

Fig. 8 – Research directions and accuracies achieved in previous work

learning representations directly from the BGP graph during the training process, eliminating the need for statistical features or graph features. However, this approach was still limited by the supervised learning of the graph representations, which requires carefully labelled BGP data and restricts the generalization of the model.

The work presented in this paper addresses this issue through the use of an unsupervised GAE to learn the graph representations. This approach:

- enhances the generalization capability of the model,
- does not require labelled BGP data which allows the use of a large amount of unlabelled BGP data,
- uses every node of the graph to capture all the subtlety of BGP,
- provides an 0.99 accuracy for BGP anomaly detection on large-scale events.

The presented results are also promising for small-scale BGP anomaly detection such as origin hijacking and path hijacking which will be explored in future works.

REFERENCES

[1] Bahaa Al-Musawi, Philip Branch, and Grenville Armitage. “BGP Anomaly Detection Techniques: A Survey”. In: *IEEE Communications Surveys & Tutorials* 19.1 (Feb. 2017), pp. 377–396. ISSN: 1553-877X. DOI: 10.1109/comst.2016.2622240. URL: http://dx.doi.org/10.1109/comst.2016.2622240.

[2] Ratul Mahajan, David Wetherall, and Tom Anderson. “Understanding BGP misconfiguration”. In: *ACM SIGCOMM Computer Communication Review* 32.4 (2002), pp. 3–16.

[3] Lan Wang, Xiaoliang Zhao, Dan Pei, Randy Bush, Daniel Massey, Allison Mankin, S Felix Wu, and Lixia Zhang. “Observation and analysis of BGP behavior under stress”. In: *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. 2002, pp. 183–195.

[4] S. Deshpande, T. Ho, M. Thottan, and B. Sikdar. “An Online Mechanism for BGP Instability Detection and Analysis”. In: *IEEE Transactions on Computers* 58.11 (Nov. 2009), pp. 1470–1484. ISSN: 1557-9956. DOI: 10.1109/TC.2009.91.

[5] James H Cowie, Andy T Ogielski, B Premore, Eric A Smith, and Todd Underwood. “Impact of the 2003 blackouts on Internet communications”. In: *Preliminary Report, Renesys Corporation (updated March 1, 2004)* (2003).

[6] RIPE. *Routing Information Service (RIS)*. URL: https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/routing-information-service-ris (visited on 07/20/2020).

[7] RouteViews. *Routeviews - University of Oregon Route Views Project*. en-US. URL: http://www.routeviews.org/routeviews/ (visited on 07/20/2020).

[8] Qingye Ding, Zhida 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. 003352–003355. DOI: 10.1109/SMC.2016.7844751.

[9] 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. DOI: 10.1109/INISTA.2011.5946083.

[10] Min Cheng, Qing Li, Jianming Lv, Wenyin Liu, and Jianping Wang. "Multi-scale LSTM model for BGP anomaly classification". In: *IEEE Transactions on Services Computing* (2018).

[11] Kevin Hoarau, Pierre-Ugo Tournoux, and Tahiry Razafindralambo. "Suitability of Graph Representation for BGP Anomaly Detection". In: *2021 IEEE 46th Conference on Local Computer Networks (LCN) (LCN 2021)*. Edmonton, Canada, Oct. 2021.

[12] Odnan Ref Sanchez, Simone Ferlin, Cristel Pelsser, and Randy 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.

[13] Kevin Hoarau, Pierre Ugo Tournoux, and Tahiry Razafindralambo. "BGNN: Detection of BGP Anomalies Using Graph Neural Networks". In: *2022 IEEE Symposium on Computers and Communications (ISCC)*. IEEE. 2022, pp. 1–6.

[14] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[15] Thomas N Kipf and Max Welling. "Variational graph auto-encoders". In: *arXiv preprint arXiv:1611.07308* (2016).

[16] Lixin Gao. "On inferring autonomous system relationships in the Internet". In: *IEEE/ACM Transactions on Networking* 9.6 (Dec. 2001), pp. 733–745. ISSN: 1063-6692. DOI: 10.1109/90.974527.

[17] S. Hares Y. Rekhter T. Li. *A Border Gateway Protocol 4 (BGP-4)*. RFC 4271. IETF, 2006. URL: https://tools.ietf.org/html/rfc4271.

[18] Kevin Butler, Toni R Farley, Patrick McDaniel, and Jennifer Rexford. "A survey of BGP security issues and solutions". In: *Proceedings of the IEEE* 98.1 (2009), pp. 100–122.

[19] Kevin Hoarau, Pierre-Ugo Tournoux, and Tahiry 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, June 2021.

[20] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. "A comprehensive survey on graph neural networks". In: *arXiv preprint arXiv:1901.00596* (2019).

[21] Matthias Fey and Jan E. Lenssen. "Fast Graph Representation Learning with PyTorch Geometric". In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019.

[22] Lutz Prechelt. "Early stopping—but when?" In: *Neural networks: tricks of the trade: second edition* (2012), pp. 53–67.

[23] Claude Sammut and Geoffrey I Webb. *Encyclopedia of machine learning*. Springer Science & Business Media, 2011.

[24] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George 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.

[25] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. “DeepGCNs: Can GCNs Go as Deep as CNNs?” In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 9267–9276.

[26] Kevin Hoarau, Pierre Ugo Tournoux, and Tahiry Razafindralambo. “Detecting forged AS paths from BGP graph features using Recurrent Neural Networks”. In: *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE. 2022, pp. 735–736.

[27] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. “Temporal graph networks for deep learning on dynamic graphs”. In: *arXiv preprint arXiv:2006.10637* (2020).

AUTHORS

Kevin Hoarau received his MSc in computer science from University of Reunion Island (UR) in 2017 and his PhD degree in computer science from UR in 2022. He is currently a contract teacher at University of Reunion Island. His research interests include machine learning, deep learning and graph neural networks for networking.

Pierre Ugo Tournoux received his MSc in computer science from University Pierre et Marie Curie (UPMC) in 2007 and his PhD degree in computer science from LAAS-CNRS in 2010. He his currently an associate professor at University of Reunion Island. His research interests include Internet of things and privacy issues in wireless networks.

Tahiry Razafindralambo received his MSc in applied statistics and computer science from the university of Antananarivo in 2001, his PhD degree in computer science from the INSA de Lyon in 2007 and his HDR in 2013 from University of Lille. He is currently an associate professor at University of Reunion Island. His research interests are mainly focused on distributed algorithms and protocols design for network communication and performance evaluation.