Detection of contextual anomalies in attributed graphs.

Rémi Vaudaine Baptiste Jeudy Christine Largeron

Univ Lyon, UJM-Saint-Etienne, CNRS, Institut d Optique Graduate School
Laboratoire Hubert Curien UMR 5516, F-42023, SAINT-ETIENNE, France
{remi.vaudaine,baptiste.jeudy,christine.largeron}@univ-st-etienne.fr

Abstract

Graph anomaly detection have proved very useful in a wide range of domains. For instance, for detecting anomalous accounts (e.g. bots, terrorists, opinion spammers or social malwares) on online platforms, intrusions and failures on communication networks or suspicious and fraudulent behaviors on social networks. However, most existing methods often rely on pre-selected features built from the graph, do not necessarily use local information and do not consider context based anomalies. To overcome these limits, we present CoBaGAD, a Context-Based Graph Anomaly Detector which exploits local information to detect anomalous nodes of a graph in a semi-supervised way. We use Graph Attention Networks (GAT) with our custom attention mechanism to build local features, aggregate them and classify unlabeled nodes into normal or anomaly. Finally, we show that our algorithm is able to detect anomalies with high precision and recall and, outperforms state-of-the-art baselines.

Keywords : Graph neural network, Graph anomaly detection, Node classification.

1 Introduction

Anomaly detection has been a field of intense research for the last decades, both for graph data [3] and for vector data [24]. According to these last ones, anomalies are substantial variations from the norm. In a graph, a node can be an anomaly because of its neighborhood, its attributes or a combination of both. For instance, in a graph with community structure (i.e. containing sets of highly connected nodes) an anomaly can correspond to a node which do not really belong to any community either because it is isolated or because it forms a bridge between two groups. In an attributed graph with assortativity, it can be a node whose attributes are significantly different from those of its neighbors.

[0]Supplementary Information available online: https://github.com/vaudaine/Detection-of-contextual-anomalies-in-attributed-graphs

In this paper, we introduce and study a new particular case of node anomaly: context-based anomaly. This kind of anomaly is relatively frequent in practice. For instance, in a bibliographic network where the nodes correspond to papers assigned to thematic categories and there is a link from a document node to another if the first one cites the second, a contextual anomaly can correspond to a node belonging to the category 'Fruit' (because it contains the word 'Apple') which is cited by documents belonging to the category 'computer science'. Context-based anomalies are also often associated with fraud or corruption. In those situations, experts try to use "patterns" or contexts to find these fraud or corruption cases. For instance, a company has a CEO or an accountant who has an account in a tax haven then this company is more likely to be fraudulent. To define this kind of anomaly, we consider that there exists an unknown small subgraph (a context), and a distinguished node in this subgraph, such that each time this subgraph occurs in the data, then the node corresponding to the distinguished node is an anomaly with a high probability p. In this paper, for simplicity, we consider the case $p = 1$. We call this unknown subgraph a context and the corresponding anomalies "context-based anomalies". We argue that these context-based anomalies are interesting and, as the experiments show, not always well detected by current approaches.

Their detection can be tackled in a supervised, semi-supervised or unsupervised way. The first one supposes that a training sample consisting of instances of the two classes (normal nodes and anomaly nodes) is available. In the second one, a set of unlabeled instances is also exploited during training while in the latter one, no labels are available. In this paper, we consider the semi-supervised case. More precisely, we use a transductive setting: the data consist of a single graph for which a proportion of the nodes is labeled (either "anomaly" or "normal") and other nodes have no labels. The goal is to find the labels of the unlabeled nodes. The difference with a fully supervised setting is that the learning algorithm can use the unlabeled nodes even if it has no access to their labels which is helpful since removing them would change the connectivity of the graph and thus disrupt the learning.

Generally, node anomaly detection is addressed by finding a representation of the nodes in a feature space and then identifying anomalies in this space. The features can be hand-made or automatically learned, for instance by a Graph Neural Network (GNN) which is the state of the art for node classification and more generally for solving many supervised or unsupervised problems on graphs. In this paper, we propose to use this approach to automatically and simultaneously learn a suited representation of the nodes and detect the anomalies. More precisely, we propose CoBaGAD (Context-Based Graph Anomaly Detector) a semi-supervised algorithm to detect graph anomalies. CoBaGAD is a variation of Graph Attention Networks (GAT) where the attention mechanism has been changed by a custom one allowing better feature selection.

To carry out an experimental evaluation of CoBaGAD, we need datasets. However, to our knowledge, real data for this kind of anomaly is not publicly available. Thus, we used synthetic and real graph data in which we artificially introduced context-based anomalies.

Our contributions are:

- We define a new kind of node anomaly in a graph.
- We propose a GNN architecture to detect it.
- We validate our model on several kinds of graph with different pattern anomalies and compare it with GAT, GCN, GraphSage and Node2vec + LOF.

The paper is organized as follows. First, we review related works. We define the problem in Section 3 and present our method to detect context-based anomaly in attributed graphs in Section 4. Then, we describe our evaluation protocol and the experiments carried out to evaluate the ability of our method to detect anomalies. Finally, we discuss the obtained results which are generally better than those provided by state of the art methods.

2 Related work

We can distinguish two main fields of research in anomaly detection according to the type of data investigated. First, anomaly detection with tabular data, where the elements (or instances) are described by feature vectors and which aims at finding vectors in the space that are "far" from the others. Different criteria have been proposed to quantify the notion of "far" from the others. On the other hand, graph anomaly detection deals with relational data. While they do not use the same type of data, these fields share common ideas. In this section, we review state-of-the-art methods for both types of data and, finally, for graph mining. Both fields have variations as static and dynamic anomaly detection. Thereafter, we will focus only on static data.

2.1 Anomaly detection with vector data

In the literature, a large number of methods have been proposed to solve the task of anomaly detection on vector data [17] [33]. They can be classified in several families. The first family, detailed in particular in [31], uses a probability model to describe the available instances and then identifies the anomalies as elements having a low probability of occurrence according to the model generated from the data. The second approach, based on notions of distance or neighborhood detects anomalies by looking for elements that are too far from their neighbors in the representation space or whose neighborhood is not sufficiently dense [20] [1] [34]. Among the methods which directly exploit the neighborhood, we can mention for instance [30] [20][4] and among those which estimate the density around an instance, LOF (Local Outlier Factor) [8]. We can also cite methods based on the construction of forests such as iForest [23] or iNNE [5], which consist in recursively partitioning all the instances using attributes so as to build a tree in which an instance located in a leaf far from the root is more likely to be an anomaly than a less distant instance. However, comparative

studies of these methods have highlighted difficulties that can lead to a decrease in their performance, in particular the curse of dimensionality when the number of attributes is high or even the imbalance of the data.

2.2 Anomaly detection with graph data

Concerning anomaly detection with graph data, the methods proposed in the literature rarely aim at finding the same type of anomalies but they can be classified into different categories depending on the kind of graph that they deal with. The first kind is plain graph which is a graph without features. Scan [39] and PAICAN [6] are graph clustering algorithms that detect anomalies as byproducts. Such anomalies are usually bridges between communities as in [37]. Structural anomalies such as anomalous edges or irregular subgraphs can be detected by Autopart [10], [27] and [12]. For its part, OddBall [2] is an approach based on power laws which detects elements of the graph whose characteristics deviate from these power laws, for instance nodes with very high degree. By this way, this method can detect near stars, near cliques, dominant edges and heavy vicinity in a weighted graph. On the other hand, some methods [26] [28] [29] have been proposed to deal with attributed graphs. FocusGo [29] is a method that focuses on user-specific attributes to cluster nodes that are similar. Outliers are defined as nodes which structurally belong to a cluster but deviate from its focus attributes. Recently, an extension [28] of this method has been proposed. It is able to both find communities and extract local information (find focus features). Anomalies are nodes or groups of nodes that cannot be easily summarized by a community and some focus features. GOutrank [26] and ConOut [32] are outlier ranking techniques that find subgraphs as context of nodes and subspaces of their features to focus on those that deviate from these subspaces. Thus, like in [28] [29], anomalies are nodes whose features deviate a lot from those of their neighbors.

2.3 Machine learning and graph mining

In recent years, machine learning has become more popular to mine graphs. In particular, the rise of Word2vec [25] to deal with textual data has also led to a renewal of graph embedding which aims to project a graph into a low-dimensional space such that each node is represented by a vector [13] [41] [9]. In the context of anomaly detection, any method that deals with vector data can be used afterwards. On the other hand, some embedding algorithms [40] [18] have been created to both represent the nodes as vectors and find outliers. However, they are unsupervised whereas we focus on semi-supervised learning.

Today, state-of-the-art methods such as SDNE [36], GraphSAGE [15], GCN [19] or GAT [35] [7] use deep learning to mine graphs and they have notably obtained very good performance on tasks such as node classification as shown in recent surveys [38] [11]. But, to the best of our knowledge, these deep learning methods have not been used in anomaly detection context. Thus, in this work,

we continue down this path by proposing a deep learning based method to classify nodes into two classes: anomalies and normal nodes in a semi-supervised context.

3 Problem definition

We aim at detecting contextual anomalies which are nodes of the graph whose local context exhibits a singular arrangement.

More formally, let $G(V, E, X)$ be a graph on a set of n nodes $V = \{v_i\}$, a set of edges $E = \{e_{ij}\}$ and a feature matrix $X \in \mathbf{R}^{n \times F}$. Each row $\vec{x}_i$ in matrix X is the feature vector of node v_i. We consider that there exists an unknown small subgraph (a context), and a distinguished node in this subgraph, such that each time this subgraph occurs in the data, then the node corresponding to the distinguished node is an anomaly (we give several examples in sec. 5.1).

We use a transductive setting: the data consists of a single graph for which a proportion of the nodes is labeled (either "anomaly" or "normal") and other nodes have no labels. The goal is to find the labels of the unlabeled nodes. However and most importantly, the conditions (i.e., the context) which make a node anomalous are not known during the training of the model.

4 Our method: CoBaGAD

The main idea of our method is to learn simultaneously two two-classes classifiers with attention mechanisms. Then, local information is aggregated to determine whether a node is normal or not. Parameters of the network are learnt with a standard classification loss in a semi-supervised fashion. For this, we propose to improve Graph Attention Networks (GAT)[35].

Global affine transformation: The first step is an affine transformation followed by a non linearity σ. This function σ is applied elementwise. The parameters are the matrix $W \in \mathbf{R}^{F \times F'}$ and a row vector $b \in \mathbf{R}^{1 \times F'}$. The identity matrix is denoted $\mathbb{1}$.

$$\Lambda = \sigma\left(XW + \mathbb{1}b\right) \tag{1}$$

This step transforms the features $\vec{x}_i$ independently for each node v_i. The ith row of Λ is the new representation for node v_i, and notice that since the matrix W is not necessarily square, this new representation can have more or less features than $\vec{x}_i$.

Attention layer: It consists of k attention heads (k is an hyper-parameter). For each attention head $c \in \{0, ..., k-1\}$, we perform a local linear transformation followed by a weighted aggregate: First, a local linear transformation $W_c \in \mathbf{R}^{F' \times F'}$ is applied on the features:

$$\Lambda_c = \Lambda W_c \tag{2}$$

Then, for each edge $(i, j) \in E$, the value $e_{i,j,c}$ is computed:

$$e_{i,j,c} = LeakyReLU\left((\vec{\lambda}_{i,c} \odot \vec{\lambda}_{j,c}).\vec{u}_c\right) \tag{3}$$

where $\odot$ is the Hadamard product, $\vec{\lambda}_{i,c}$ and $\vec{\lambda}_{j,c}$ are resp. the ith and jth rows of Λ_c, and $\vec{u}_c \in \mathbf{R}^{F'}$ is a parameter column vector. The Hadamard product multiplied by $\vec{u}_c$ corresponds to a weighted dot-product similarity.

The attention weights $\alpha_{i,j,c}$ are defined as a normalized version of $e_{i,j,c}$ such that for each node v_i and each head c, they are positive and sum to 1:

$$\alpha_{i,j,c} = \frac{\exp(e_{i,j,c})}{\sum_{k \in N(i)} \exp(e_{i,k,c})} \tag{4}$$

where $N(i)$ is the set of the neighbors of node v_i.

The next step is to compute for each node v_i, a convex combination of the $\vec{\lambda}_{j,c}$ for all neighbors v_j of v_i using the attention weights. These weights can be seen as the amount of information that flows between nodes. Finally, the new representation $\vec{h_i}$ of the node v_i is given by the concatenation of all the representations given by the k attention heads. i.e., each node v_i is represented by a row vector of $F'k$ features.

$$\vec{h_{i,c}} = \sum_{j \in N(i)} \alpha_{i,j,c} \vec{\lambda}_{j,c} \text{ and } \vec{h_i} = \sigma'\left(\|_{c=0}^{k-1} \vec{h_{i,c}}\right) \tag{5}$$

where σ' is an activation function and $\|$ is vector concatenation.

The network can be a stack of several such attention layers, the output of each layer being the input of the next layer.

Classification: To detect anomalies, we consider that each of the head in the last layer is a 2-classes classifier (thus each $\vec{h_{i,c}} \in \mathbf{R}^2$) and we combine these classifiers by taking the argmax. i.e., if the maximum component in vector $\vec{h_i}$ is in an odd index, v_i is classified as an anomaly. If the maximum is in an even index, then it is a normal node.

The parameters that must be learnt are: W, b, and for each of the k attention heads in each attention layer: the matrix W_c and the vector $\vec{u_c}$. The hyper-parameters are F', the number of attention layers, and the number k of heads in each attention layer. The activation functions σ and σ' can also be chosen by the user.

CoBaGAD differs from GAT by two major points. First, we added a global affine transformation (Eq.1) to embed the original features in a new space. This operation improves the ability to detect anomalies and allows to reduce the dimension of the problem. Then, we use a custom attention mechanism in Eq.3. Rather than concatenating the new representations of a pair of nodes, we compute a similarity between them with the Hadamard product. The attribute weight vector $\vec{u}_c$ focuses on most important part of this product for classification.

5 Experiments

5.1 Dataset generation and description

As benchmarks corresponding to the kind of anomaly considered in this paper are not publicly available, to experimentally evaluate our model and compare it with the state of the art, we have artificially introduced context-based anomalies in many different graphs, real or synthetic. Table 1 gives the name, the number of nodes and the number of edges of these graphs. G_0 is a random Erods-Renyi graph. G_1 and G_4 are also generated, respectively with Dancer [22] and LFR [21], which mimic real-world graph's behaviour. Moreover, we chose real world graphs that are common in the literature: Polblogs [1], Cora [2] and Facebook [3].

The following process has been applied to transform each of these graphs $G(V, E)$ into an attributed graph $G(V, E, X)$. The feature vector $\vec{x_i}$ of node v_i is defined as a one-hot vector of dimension 5. In our illustrative example related to fraud detection, such features can be interpreted as the role in a company (*e.g.* $[0, 0, 1, 0, 0]$ represents the CEO and $[0, 1, 0, 0, 0]$ represents an employee). In the following, these one-hot vectors are flagged as colors: blue (B), green (G), red (R), yellow (Y) and purple (P).

Among the nodes of the graph, a few percent (between 4% and 6%) are flagged as anomalies if they follow a simple rule described by a context. These rules are presented in Table 2. In this table, $\bar{Y}$ means that anomalies are nodes that have colour $Yellow$ whereas B means that anomalies have at least one neighbor whose colour is $Blue$. For example, A_6 represent nodes that have the colour yellow (Y) or have at least one neighbor with colour blue (B) and at least one neighbor with colour yellow (Y).

Note that our algorithm CoBaGAD does not know how the anomalies have been created. Indeed, in a real-world case, the expert would flag some nodes as anomalies but he does not necessary know the conditions which make a node anomalous. Thus the goal of our algorithm is to recognize these anomalies without this contextual knowledge.

5.2 Experimental setup

All nodes of the graphs belong to either the set of anomalies or the set of normal nodes. In a transductive setup, nodes are split into train, validation and test sets. The train set is made of 50% of the total anomalies. Then, we add as many normal nodes as there are anomalies. The same applies for validation set with 25% of anomalies. The test set is composed of the remaining 25% of anomalies and 25% of normal nodes of the graph. Balancing train and validation sets in order to have as many normal nodes as anomalies improved a lot the results.

[1] http://konect.cc/networks/dimacs10-polblogs/
[2] https://relational.fit.cvut.cz/dataset/CORA
[3] https://snap.stanford.edu/data/egonets-Facebook.html

Graph	Name	Nodes	Edges
G_0	Erdos-Renyi	10000	24907
G_1	Dancer	10000	189886
G_2	Facebook	4039	88234
G_3	Polblogs	1224	16715
G_4	LFR	1000	5622
G_5	Cora	1433	5429

Table 1: Datasets characteristics: name of the graphs, number of nodes and edges.

Anomalies	Definition
A_0	$B \wedge G$
A_1	$(B \wedge G) \vee (B \wedge R)$
A_2	$(B \wedge G) \vee (R \wedge Y)$
A_3	$\bar{Y}$
A_4	$\bar{Y} \wedge B$
A_5	$\bar{Y} \wedge B \wedge Y$
A_6	$\bar{Y} \vee (B \wedge Y)$

Table 2: Anomalies characteristics. B = blue, G = green, R = red, Y = yellow, P = purple. $\bar{Y}$ means that anomalies are nodes that have colour Y. B means that anomalies have at least one neighbor whose colour is B.

Thus a part of negative examples (normal nodes) is ignored during training. Due to space limitations, we only show the results using this sampling strategy. To ensure the reproducibility of our results, code and datasets are available in our GitHub [4].

We compare our algorithm, CoBaGAD, with state-of-the-art methods in node classification: Graph Convolution Networks [19] (GCN), Graph Attention Networks [35] (GAT), GraphSAGE [15] with mean aggregator and an unsupervised anomaly detection approach based on Node2vec [14] and LOF [8]. For every deep learning method, we learn a single layer. Given the kind of studied pattern, add more layers seems not relevant. For CoBaGAD, we use GELU [16] as activation function σ and softmax as activation function σ', $F' = 2$, $k = 2$ as we learn two 2-classes classifiers for both anomalies and normal nodes. For GAT and GraphSAGE, we use the same parameters. For GCN, we use *localpool* filter, softmax as activation function and output of dimension 2. For every algorithm, we tried two versions: without self-loop and with self-loops by adding the identity matrix to the adjacency matrix and we present the best results. We train for 1000 epochs with Adam optimizer and a learning rate of $5e-3$ on the train set and validate it at each step. The weights of the networks are kept when the accuracy on the validation set is the highest. We use the standard categorical

[4] https://github.com/vaudaine/Detection-of-contextual-anomalies-in-attributed-graphs

cross-entropy loss: $L(Y^{true}, Y^{pred}) = -\sum_{j=1}^{k}\sum_{i}^{N}(y_{ij}^{true} \times log(y_{ij}^{pred}))$. Concerning the unsupervised approach, it is an association of Node2vec embedding and LOF anomaly detection. First, we compute an embedding of the graph using Node2vec with $p = 1, q = 1$ and dimension 128. It outputs a new representation for every node v_i. This representation is concatenated with its feature vector $\vec{x_i}$. Finally, these vectors are fed to a LOF classifier in order to detect anomalies.

6 Results

For each model, each dataset and each type of anomaly, experiments are conducted 12 times by changing the train/validation/test split. We choose the 3 best results on the validation and report the mean and standard deviation of precision obtained on the test set. These results can be found in Table 3. Due to space constraints, we did not report results for the unsupervised approach combining Node2vec and LOF since the precision was very low as expected (between 0% and 20%). Also, the recall on anomalies, and the recall and precision on normal nodes are not reported. They are indeed very high for all graphs and type of anomaly (most of the time above 98%) and no significant differences can be observed between the different models. These results are however available in additional materials [5].

The results show that our algorithm achieves state-of-the-art performance across all datasets and anomalies. More specifically, for A_0, A_1 and A_2 which all are anomalies based on the pattern $B \wedge G$, our method always outperforms the other competitors (except for A_0, G_5 where it is still very relevant). We are able to improve upon GAT, our principal contendor, by at least 2% on A_0, G_0 up to 62% on A_0, G_2. Attention based methods are better than the others (GCN, GraphSage, Node2vec + LOF) when dealing with these types of anomaly.
Anomalies A_3 to A_6 rely on the pattern $\bar{Y}$ which means that the considered nodes are yellow. This means that the information about the node itself is required. A_3 is a very simple pattern where anomalies are defined by the simplest pattern: nodes are just yellow. In that case, we can suppose that it is easy for many algorithms to perform well on detecting those nodes. In fact, GraphSAGE shows good performance for most of the graphs but lack a bit of consistency. GCN is more consistent but results are worse than those provided by GraphSAGE. While GAT fails to show good performance, our method is the most consistent and show very good results in general. Then, for A_4 to A_6, as the pattern becomes more complex, CoBaGAD remains the only method that, except a few cases, correctly detects the anomalies.

7 Conclusion

We have defined a new kind of anomaly based on a context. Such anomalies follow a simple pattern. Then, we have presented Context Based Graph

[5] https://github.com/vaudaine/Detection-of-contextual-anomalies-in-attributed-graphs

A_0	G_0	G_1	G_2	G_3	G_4	G_5
CoBaGAD	$\mathbf{0.98 \pm 0.03}$	$\mathbf{0.96 \pm 0.03}$	$\mathbf{0.95 \pm 0.03}$	$\mathbf{0.87 \pm 0.09}$	$\mathbf{0.85 \pm 0.13}$	0.92 ± 0.12
GAT	0.96 ± 0.04	0.83 ± 0.02	0.33 ± 0.08	0.55 ± 0.08	0.59 ± 0.13	$\mathbf{0.93 \pm 0.1}$
GCN	0.34 ± 0.02	0.12 ± 0.01	0.11 ± 0.0	0.18 ± 0.02	0.31 ± 0.03	0.26 ± 0.02
GraphSage	0.53 ± 0.01	0.56 ± 0.03	0.63 ± 0.05	0.44 ± 0.08	0.36 ± 0.08	0.52 ± 0.03

A_1	G_0	G_1	G_2	G_3	G_4	G_5
CoBaGAD	$\mathbf{0.96 \pm 0.03}$	$\mathbf{0.98 \pm 0.02}$	$\mathbf{0.78 \pm 0.1}$	$\mathbf{0.73 \pm 0.05}$	$\mathbf{0.72 \pm 0.15}$	$\mathbf{0.85 \pm 0.12}$
GAT	0.74 ± 0.3	0.55 ± 0.21	0.51 ± 0.14	0.63 ± 0.07	0.46 ± 0.19	0.8 ± 0.18
GCN	0.34 ± 0.02	0.19 ± 0.03	0.13 ± 0.02	0.19 ± 0.03	0.34 ± 0.04	0.26 ± 0.01
GraphSage	0.46 ± 0.04	0.5 ± 0.02	0.51 ± 0.06	0.39 ± 0.06	0.38 ± 0.07	0.47 ± 0.03

A_2	G_0	G_1	G_2	G_3	G_4	G_5
CoBaGAD	$\mathbf{0.61 \pm 0.04}$	$\mathbf{0.62 \pm 0.06}$	$\mathbf{0.58 \pm 0.2}$	$\mathbf{0.47 \pm 0.14}$	$\mathbf{0.64 \pm 0.15}$	$\mathbf{0.72 \pm 0.06}$
GAT	0.52 ± 0.02	0.42 ± 0.1	0.29 ± 0.03	0.35 ± 0.06	0.3 ± 0.03	0.51 ± 0.11
GCN	0.27 ± 0.02	0.12 ± 0.0	0.14 ± 0.01	0.2 ± 0.04	0.25 ± 0.02	0.23 ± 0.03
GraphSage	0.33 ± 0.01	0.38 ± 0.02	0.44 ± 0.03	0.44 ± 0.05	0.33 ± 0.03	0.38 ± 0.01

A_3	G_0	G_1	G_2	G_3	G_4	G_5
CoBaGAD	0.99 ± 0.01	$\mathbf{0.99 \pm 0.01}$	0.95 ± 0.05	$\mathbf{0.91 \pm 0.13}$	0.89 ± 0.15	0.93 ± 0.02
GAT	0.82 ± 0.07	0.79 ± 0.03	0.78 ± 0.07	0.55 ± 0.03	0.43 ± 0.01	0.61 ± 0.09
GCN	0.95 ± 0.02	0.97 ± 0.02	0.91 ± 0.07	0.8 ± 0.21	$\mathbf{0.94 \pm 0.08}$	0.78 ± 0.04
GraphSage	$\mathbf{1.0 \pm 0.0}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{0.97 \pm 0.04}$	0.71 ± 0.1	0.68 ± 0.24	$\mathbf{0.97 \pm 0.03}$

A_4	G_0	G_1	G_2	G_3	G_4	G_5
CoBaGAD	$\mathbf{0.9 \pm 0.1}$	$\mathbf{0.95 \pm 0.04}$	$\mathbf{0.97 \pm 0.04}$	$\mathbf{0.8 \pm 0.07}$	0.62 ± 0.1	$\mathbf{0.89 \pm 0.09}$
GAT	0.5 ± 0.07	0.77 ± 0.15	0.77 ± 0.03	0.67 ± 0.23	0.51 ± 0.1	0.43 ± 0.05
GCN	0.44 ± 0.01	0.7 ± 0.03	0.71 ± 0.12	0.6 ± 0.12	$\mathbf{0.65 \pm 0.07}$	0.35 ± 0.02
GraphSage	0.46 ± 0.03	0.73 ± 0.02	0.72 ± 0.02	0.58 ± 0.11	0.61 ± 0.04	0.41 ± 0.03

A_5	G_0	G_1	G_2	G_3	G_4	G_5
CoBaGAD	$\mathbf{0.84 \pm 0.02}$	$\mathbf{0.9 \pm 0.03}$	$\mathbf{0.82 \pm 0.07}$	$\mathbf{0.61 \pm 0.03}$	0.51 ± 0.23	$\mathbf{0.84 \pm 0.22}$
GAT	0.69 ± 0.11	0.74 ± 0.07	0.71 ± 0.09	0.57 ± 0.07	0.52 ± 0.09	0.46 ± 0.1
GCN	0.35 ± 0.01	0.67 ± 0.04	0.62 ± 0.07	0.6 ± 0.04	0.46 ± 0.13	0.23 ± 0.02
GraphSage	0.34 ± 0.01	0.68 ± 0.0	0.68 ± 0.06	0.51 ± 0.04	$\mathbf{0.56 \pm 0.06}$	0.3 ± 0.04

A_6	G_0	G_1	G_2	G_3	G_4	G_5
CoBaGAD	$\mathbf{0.9 \pm 0.01}$	$\mathbf{0.89 \pm 0.05}$	$\mathbf{0.54 \pm 0.27}$	$\mathbf{0.62 \pm 0.2}$	0.26 ± 0.09	0.4 ± 0.04
GAT	0.55 ± 0.12	0.59 ± 0.12	0.39 ± 0.05	0.48 ± 0.1	0.27 ± 0.03	0.39 ± 0.03
GCN	0.38 ± 0.05	0.26 ± 0.09	0.2 ± 0.07	0.21 ± 0.01	0.32 ± 0.13	0.46 ± 0.11
GraphSage	0.67 ± 0.04	0.58 ± 0.05	0.48 ± 0.06	0.49 ± 0.16	$\mathbf{0.48 \pm 0.06}$	$\mathbf{0.65 \pm 0.14}$

Table 3: Precision of the detection of anomalies A_0-A_6 on several graphs (G_0-G_5) in the testing set. **Bold**: best in column.

Anomaly Detector, CoBaGAD, an extension of the Graph Attention Networks that focuses on detecting those anomalies. Through intensive transductive ex-

periments, we demonstrate the ability of our method to identify such pattern anomalies and to outperform state-of-the-art algorithms.
Different improvements can be addressed as future work such as scoring anomalies instead of binary classifying. Another particularly interesting field of research in the domain of anomaly detection is the interpretability of the detected anomalies. The objective would be to be able to recover the context that defines anomalies. Then, we will also study anomaly defined by contexts of larger diameter. This will involve using networks with more layers to increase the "field of view".

Acknowledgement

This work has been supported by IDEXLYON ACADEMICS Project ANR-16-IDEX-0005 of the French National Research Agency.

References

[1] Aggarwal, C.C.: Outlier Analysis. Springer International Publishing (2017)

[2] Akoglu, L., McGlohon, M., Faloutsos, C.: Oddball: Spotting Anomalies in Weighted Graphs. In: PAKDD. pp. 410–421 (2010)

[3] Akoglu, L., Tong, H., Koutra, D.: Graph based anomaly detection and description: a survey. Data Min. Knowl. Discov. **29**(3), 626–688 (2015)

[4] Angiulli, F., Pizzuti, C.: Fast outlier detection in high dimensional spaces. In: PKDD. pp. 15–26 (2002)

[5] Bandaragoda, T.R., Ting, K.M., Albrecht, D., Liu, F.T., Zhu, Y., Wells, J.R.: Isolation-based anomaly detection using nearest-neighbor ensembles. Computational Intelligence **34**(4), 968–998 (2018)

[6] Bojchevski, A., Günnemann, S.: Bayesian robust attributed graph clustering: Joint learning of partial anomalies and group structure. In: AAAI Conference on Artificial Intelligence. pp. 2738–2745 (2018)

[7] Bresson, X., Laurent, T.: Residual gated graph convnets. In: ICLR (2018)

[8] Breunig, M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: Identifying density-based local outliers. In: ICMD. pp. 93–104 (2000)

[9] Cai, H., Zheng, V., Chen-Chuan Chang, K.: A comprehensive survey of graph embedding: Problems, techniques, and applications. TKDE **30**(9), 1616–1637 (2018)

[10] Chakrabarti, D.: AutoPart: Parameter-Free Graph Partitioning and Outlier Detection. In: PKDD. pp. 112–124 (2004)

[11] Dwivedi, V.P., Joshi, C.K., Laurent, T., Bengio, Y., Bresson, X.: Benchmarking graph neural networks (2020), https://arxiv.org/abs/2003.00982

[12] Eberle, W., Holder, L.: Discovering Structural Anomalies in Graph-Based Data. In: ICDMW 2007. IEEE (2007)

[13] Goyal, P., Ferrara, E.: Graph embedding techniques, applications, and performance: A survey. Knowledge-Based Systems **151**, 78–94 (2018)

[14] Grover, A., Leskovec, J.: Node2vec: Scalable feature learning for networks. In: SIGKDD. pp. 855–864. ACM (2016)

[15] Hamilton, W.L., Ying, R., Leskovec, J.: Inductive Representation Learning on Large Graphs. In: NeurIPS. vol. 30, pp. 1024–1034 (2017)

[16] Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus) (2020), https://arxiv.org/abs/1606.08415v3

[17] Hodge, V., Austin, J.: A survey of outlier detection methodologies. Artificial Intelligence Review pp. 85–126 (2004)

[18] Hu, R., Aggarwal, C.C., Ma, S., Huai, J.: An embedding approach to anomaly detection. In: ICDE. pp. 385–396 (2016)

[19] Kipf, T., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)

[20] Knorr, E.M., Ng, R.T., Tucakov, V.: Distance-based outliers: Algorithms and applications. The VLDB Journal pp. 237–253 (2000)

[21] Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. Phys. Rev. E **78** (2008)

[22] Largeron, Mougel, Benyahia, Zaïane: Dancer: dynamic attributed networks with community structure generation. Knowl Inf Syst **53**, 109–151 (2017)

[23] Liu, F., Ting, K., Zhou, Z.: Isolation Forest. In: ICDM. pp. 413–422 (2008)

[24] Mehrotra, K.G., Mohan, C., Huang, H.: Anomaly Detection Principles and Algorithms. Springer International Publishing (2017)

[25] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NeurIPS. p. 3111–3119 (2013)

[26] Müller, E., Sánchez, P., Mülle, Y., Böhm, K.: Ranking outlier nodes in subspaces of attributed graphs. In: ICDEW. pp. 216–222 (2013)

[27] Noble, C.C., Cook, D.J.: Graph-based anomaly detection. In: SIGKDD. p. 631–636. Association for Computing Machinery (2003)

[28] Perozzi, B., Akoglu, L.: Discovering Communities and Anomalies in Attributed Graphs: Interactive Visual Exploration and Summarization. ACM TKDD **12**(2) (Mar 2018)

[29] Perozzi, B., Akoglu, L., Iglesias Sánchez, P., Müller, E.: Focused clustering and outlier detection in large attributed graphs. In: SIGKDD. pp. 1346–1355 (2014)

[30] Ramaswamy, S., Rastogi, R., Shim, K.: Efficient algorithms for mining outliers from large data sets. SIGMOD Rec **29(2)**, 427–438 (2000)

[31] Rousseeuw, P., Hubert, M.: Robust statistics for outlier detection. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **1(1)**, 73–79 (2011)

[32] Sánchez, P.I., Müller, E., Irmler, O., Böhm, K.: Local context selection for outlier ranking in graphs with multiple numeric node attributes. In: SSDM (2014)

[33] Su, X., Tsai, C.: Outlier detection. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **1(3)**, 264–268 (2011)

[34] Ting, K., Aryal, S., Washio, T.: Which outlier detector should i use? ICDM pp. 8–8 (2018)

[35] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph Attention Networks. International Conference on Learning Representations (2018)

[36] Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: SIGKDD. pp. 1225–1234 (2016)

[37] Wang, X., Davidson, I.: Discovering contexts and contextual outliers using random walks in graphs. In: ICDM. pp. 1034–1039 (2009)

[38] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems pp. 1–21 (2020)

[39] Xu, X., Yuruk, N., Feng, Z., Schweiger, T.A.J.: Scan: A structural clustering algorithm for networks. In: SIGKDD. pp. 824–833 (2007)

[40] Yu, W., Cheng, W., Aggarwal, C.C., Zhang, K., Chen, H., Wang, W.: NetWalk: A Flexible Deep Embedding Approach for Anomaly Detection in Dynamic Networks. In: SIGKDD. pp. 2672–2681 (2018)

[41] Zhang, D., Yin, J., Zhu, X., Zhang, C.: Network representation learning: A survey. IEEE Transactions on Big Data **6**(1), 3–28 (2020)