2024 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 22–25, 2024, LONDON, UK

DUALGCN: A CONVOLUTIONAL NETWORK FOR ANOMALY DETECTION IN DIRECTED GRAPHS BASED ON SYMMETRIZED AND SKEW-SYMMETRIZED LAPLACIANS

Theodor-Adrian Badea, Bogdan Dumitrescu

National University of Science and Technology Politehnica Bucharest
Department of Automatic Control and Computers
313 Spl. Independenţei, 060042 Bucharest, Romania

ABSTRACT

Graph machine learning techniques and notably graph neural networks (GNNs) have seen a surge in popularity due to the suitability of graphs for being the underlying data structure in a multitude of applications. Spectral graph convolutional networks (GCNs), however, seem to encounter shortfalls when it comes to directed graphs. The root cause lies in the asymmetric nature of the adjacency matrices and of the resulting Laplacians of such graphs; being ill-behaved with respect to the spectral theory. In this paper, we attempt to overcome this limitation through DualGCN: a spectral GCN approach based on the symmetrized and skew-symmetrized Laplacian matrices. To test the proposed method, we undertake the anomaly detection problem for graph-structured data and obtain better results than with other approaches.

***Index Terms*—** directed graph neural network, spectral graph convolutional network, graph machine learning, graph anomaly detection

1. INTRODUCTION

Graphs benefit from the advantage of encapsulating both structural and relational information. In many scenarios, relevant connections must be accounted for in order to obtain an adequate description of the problem. Consequently, adopting a graph as the underlying data structure to formally describe the problem is a reasonable choice. Internet, social networks, and biochemistry are only several examples to support their ubiquitous nature.

Given the versatility and popularity of graphs, classical convolutional neural networks have easily found their graph counterparts – graph convolutional networks (GCNs). Generally, they can be classified as either spatial or spectral methods. The first category is largely straightforward, with the convolution operation based on the graph topology and characterized as a neighborhood averaging with learnable weights [1, 2]. Conversely, spectral approaches [3] are more subtle and better related to the convolution notion from signal processing. They leverage the eigendecomposition of the Laplacian to define the graph convolution as an element-wise multiplication in the Fourier basis, with eigenvectors representing the Fourier modes.

While spatial GCNs, usually, can be extended with ease to directed graphs, the spectral class suffers because of the asymmetric nature of the adjacency matrix, which, ultimately, results in an asymmetric Laplacian. The inadequacy of such matrix in defining the aforementioned convolution comes from its lack of a full set of real eigenvalues, hence the characterization of *ill-behaved* from a spectral theory perspective. As [4] and [5] suggest, the challenge of spectral GCNs in case of directed graphs resides in defining a symmetric, real-valued, and positive semidefinite Laplacian matrix, with a full set of real eigenvalues and a bounded spectrum when properly normalized. Moreover, for the benefit of the problem, it should encode as much information as possible from the graph – both structural and relational.

1.1. Contribution and content

Our focus is on tackling the anomaly detection problem for graph-structured data. The proposed approach relies on leveraging two versions of the Laplacian together with a spectral graph convolutional neural network. The Laplacian matrices used are the symmetric and skew-symmetric derived from the directed graph Laplacian. The goal is to develop an efficient and easily trainable spectral graph convolutional network in the autoencoder fashion [6], which we then use to detect anomalous nodes in our working dataset, in an unsupervised manner.

We start by emphasizing in Section 1.2 several works from domain literature which influenced this paper and then we continue the discussion with a brief description of the problem and of our graph dataset in Section 1.3. The essential core notions of spectral graph theory are covered in Section 2. Then, after having established the context, in Section 3, we describe the proposed detection approach based

This work was supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS - UEFISCDI, project number PN-III-P4-PCE-2021-0154, within PNCDI III.

on a spectral GCN with the proposed Laplacians. Finally, Section 4 is dedicated to analyzing and comparing the experimental results and Section 5 to providing some closing thoughts regarding the discussed method.

1.2. Related work

Spectral GCNs have been introduced in [3], but the computational complexity in their original form limited the applicability to small graphs. The computational issues are solved in [7] by a parametrization of the convolutional filters through a truncated expansion of Chebyshev polynomials evaluated at a scaled Laplacian $\mathcal{L}_s = \frac{2}{\lambda_{max}}\mathcal{L} - \mathcal{I}_N$. Further simplifications are brought in [8], considering $\lambda_{max} = 2$ and using a first order polynomial. Spectral convolutional networks for directed graphs are targeted in [4] and [5] with an alternative Hermitian Laplacian matrix, called magnetic Laplacian. Similarly, [9] derives another symmetric version for the Laplacian of directed graphs. In [10], three symmetric matrices are extracted and used to extend the convolution operation to directed graphs. A symmetric graph convolutional autoencoder which produces a low-dimensional latent representation from a graph is introduced in [6], targeting clustering, link prediction and visualization tasks. Other works regarding GNNs are comprehensively covered in [11], however the paper is not particularly focused on spectral GCNs.

1.3. Anomaly detection problem

The attempted task of anomaly detection in graph-structured data arises from the dataset presented in [12]. The graph is constructed from a list of money transfers among accounts provided by *Libra Internet Bank*, hence we refer to the graph using the name *Libra*. Structurally, nodes represent bank accounts and the existence of a directed edge between two nodes means that at least one transfer from source account to destination account has occurred; the weight of the edge is the total amount transferred (possibly in multiple transactions). There are no self-loops (none of the accounts is both origin and destination) and between any two nodes there could be a maximum of two edges with opposite directions, given that transfers happened both ways.

Nodes are not directly labeled as anomalous, but rather edges are. It means that a bank account is suspicious if it is involved in a suspicious transfer. This way, anomalous nodes (or alerts, as they are also called) can be part of multiple doubtful transfers and an anomaly weight can be derived to quantify the anomaly extent. The main constitutional details of the *Libra* graph are:

- number of nodes: $N = 385100$;
- number of edges: 597165;
- number of anomalous nodes: 600;
- total weight of anomalies: 1034.

Nevertheless, a more comprehensive description of this dataset and of the targeted anomaly detection problem (originated from money laundering detection) can be found in [12].

2. GRAPH FOURIER DOMAIN AND CONVOLUTIONS

This section is dedicated to emphasizing several essential notions of graph theory, aimed at the spectral graph convolution operation. We start by considering the generic and well-behaved case of an undirected graph. Then, we move towards the graphs of interest, directed and weighted, and underline their deficits w.r.t. spectral GCNs.

Let $\mathcal{G} = (\mathcal{V},\ \mathcal{E})$, with $|\mathcal{V}| = N$, be an undirected graph, where $\mathcal{V}$ and $\mathcal{E}$ are the sets of vertices and edges, respectively. Since $\mathcal{G}$ is undirected, $\mathcal{E}$ consists of unordered pairs of elements from $\mathcal{V}$. The value w_{uv} associated with such an unordered pair $(u, v) \in \mathcal{E}$ is the weight of the edge between vertices u and v. We enforce that all weights are positive. The corresponding adjacency matrix $\mathcal{A}$ of $\mathcal{G}$ is

$$\mathcal{A}(u, v) = \begin{cases} w_{uv}, & \text{if } (u, v) \in \mathcal{E}. \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

From $\mathcal{A}$, we can easily construct the diagonal degree matrix $\mathcal{D}$ of $\mathcal{G}$ as

$$\mathcal{D} = \mathrm{diag}(\mathcal{A}\,\mathbf{1}), \tag{2}$$

where $\mathbf{1}$ is the all-ones vector. One can notice that the entries of $\mathcal{D}$ are not degrees, but instead are sums of edge weights for the case of weighted graphs.

The Laplacian matrix is a fundamental graph operator and its most popular form is based on the combinatorial definition $\mathcal{L} = \mathcal{D} - \mathcal{A}$. However, a more suitable alternative for graph machine learning tasks is the symmetrically normalized Laplacian proposed in [13]: $\mathcal{L}_s = \mathcal{D}^{-\frac{1}{2}}\mathcal{L}\mathcal{D}^{-\frac{1}{2}}$.

The symmetrically normalized adjacency, $\mathcal{D}^{-\frac{1}{2}}\mathcal{A}\mathcal{D}^{-\frac{1}{2}}$, has eigenvalues in $[-1, 1]$. Accordingly, the resulting Laplacian has eigenvalues in $[0, 2]$.

In [8], authors suggest the so-called *renormalization trick*, motivated by the desire to alleviate possible exploding or vanishing gradients when applying the Laplacian operator repeatedly. Thus, it is advised to start from the adjacency matrix with added self-loops $\tilde{\mathcal{A}} = \mathcal{A} + \mathcal{I}_N$ and compute the scaled [8] Laplacian matrix

$$\tilde{\mathcal{L}}_s = \tilde{\mathcal{D}}^{-\frac{1}{2}}\tilde{\mathcal{A}}\tilde{\mathcal{D}}^{-\frac{1}{2}}, \tag{3}$$

where $\tilde{\mathcal{D}}$ is defined like in (2) with $\tilde{\mathcal{A}}$ instead of $\mathcal{A}$.

More about the impact of adding self-loops can be found in [14], where the authors focus particularly on GNNs. Evidently, both ways of defining the Laplacian, $\mathcal{L}$ and $\mathcal{L}_s$, yield symmetric matrices and, moreover, the positive nature of the weights enforced earlier makes the Laplacian positive semidefinite.

The Fourier transform of x, $\hat{x} \in \mathbb{R}^N$, and its inverse, which enable spectral filtering operations [7], are defined as

$$\begin{cases} \hat{x} = U^T x \\ x = U\hat{x}, \end{cases} \tag{4}$$

with $\mathcal{L} = U\Lambda U^T$, where U (Fourier basis) is the complete set of orthonormal eigenvectors of the Laplacian and Λ is the diagonal matrix of eigenvalues (graph frequencies) – which are real and nonnegative.

In [7], the construction of a convolutional filter through a polynomial parametrization is proposed. Consequently, polynomial filters of order K of the Laplacian are K-localized. In [15], the filter is expressed as a truncated expansion of Chebyshev polynomials:

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}), \tag{5}$$

where $T_k(t) = 2tT_{k-1}(t) - T_{k-2}(t)$, with $T_0 = 1$ and $T_1(t) = t$, is the Chebyshev polynomial of order k and $\theta \in \mathbb{R}^K$ is a vector of filter coefficients. The graph signal filtering operation, denoted $*_\mathcal{G}$, becomes

$$g_\theta *_\mathcal{G} x = g_\theta\left(\tilde{\mathcal{L}}_s\right) x = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\mathcal{L}}_s) x, \tag{6}$$

where $T_k(\tilde{\mathcal{L}}_s) \in \mathbb{R}^{N\times N}$ is the polynomial filter.

3. PROPOSED APPROACH

Our problem has a directed graph as the underlying data structure. The filtering procedure presented so far, unfortunately, does not generalize because of the asymmetric nature of the adjacency matrix and, implicitly, of the Laplacian. Besides offering no guarantees regarding the existence of a full set of real eigenvalues, there are also no guarantees about their bounds for such matrices.

We are focused on detecting graph anomalies by means of a GCN architecture, hence we attempt to apply the filtering procedure described in Section 2 on directed graphs in accordance with the spectral graph theory and with minimal information loss. Our approach makes use of two Laplacian matrices, derived from the symmetrized and skew-symmetrized adjacencies:

$$\begin{aligned} \mathcal{A}_{sym} &= \frac{1}{2}\left(\mathcal{A} + \mathcal{A}^T\right), \\ \mathcal{A}_{skew} &= \frac{1}{2}\left(\mathcal{A} - \mathcal{A}^T\right). \end{aligned} \tag{7}$$

A weight of the symmetric matrix represents the average amount transferred between two nodes, while a weight of the skew-symmetric matrix represents the difference with respect to the average of the transferred amount. One may see the transformation as the application of a Haar operator to the amounts transferred between two nodes.

The Laplacian matrix constructed using $\mathcal{A}_{sym}$ is, constitutionally, equivalent to the Laplacian of an undirected graph, therefore suitable to the notions from Section 2. Similarly with (2), we build a quasi-degree diagonal matrix

$$\mathcal{D}_{skew} = \operatorname{diag}\left(|\mathcal{A}_{skew}| \cdot \mathbf{1}\right), \tag{8}$$

where $|\mathcal{A}|$ denotes the matrix whose elements are the absolute values of the elements of $\mathcal{A}$.

Proposition 1 *The eigenvalues of the normalized skew-symmetric adjacency matrix*

$$\mathcal{D}_{skew}^{-\frac{1}{2}} \mathcal{A}_{skew} \mathcal{D}_{skew}^{-\frac{1}{2}} \tag{9}$$

are purely imaginary and have subunitary absolute values.

Proof. The matrix (9) is skew-symmetric, hence it has eigenvalues on the imaginary axis. Also, it is similar with the matrix

$$\mathcal{D}_{skew}^{-\frac{1}{2}} \left(\mathcal{D}_{skew}^{-\frac{1}{2}} \mathcal{A}_{skew} \mathcal{D}_{skew}^{-\frac{1}{2}}\right) \mathcal{D}_{skew}^{\frac{1}{2}} = \mathcal{D}_{skew}^{-1} \mathcal{A}_{skew}.$$

This matrix has zeros on the diagonal and the sum of the absolute values of the elements on each row is less than 1. By Gerschgorin's circle theorem [16], its eigenvalues lie inside the unit disk. ■

As a result, the spectral properties of the normalized skew-symmetric adjacency matrix are similar to those of the symmetric one and its use in spectral GCNs should be as well-behaved as in the symmetric case.

Our anomaly detection method relies on a pair of neural networks in the autoencoder fashion [6], hence we refer to it as **DualGCN**. The two networks are identical and share the same weights and input graph signals, however the convolutional layers differ; both use the Laplacian (3), however the first network employs $\mathcal{A}_{sym}$, whereas the second network has the Laplacian based on $\mathcal{A}_{skew}$.

We feed graph signals into each of the twin networks with the goal of learning a representation to the best extent. Each layer is a filter [7] [8] constructed from a truncated expansion of Chebyshev polynomials [15]. Considering (6), the propagation rule for a layer can be generalized as

$$X^{(l+1)} = \sigma\left(\sum_{k=0}^{K-1} T_k(\tilde{\mathcal{L}}) X^{(l)} \Theta^{(l)}\right), \tag{10}$$

where $\Theta^{(l)}$ represents the learnable filter parameters of the layer and σ the nonlinearity, our choice being ReLU.

Compared to [8], which addresses the task of semi-supervised classification, we are concerned with the unsupervised approach. Our dataset has two clearly unbalanced classes, anomalies accounting for less than 0.16%

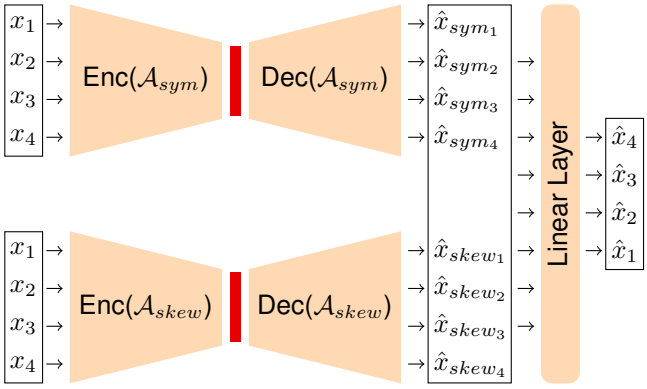

Fig. 1. DualGCN architecture with 4 input signals and concatenation linear layer, latent representation shown in red.

(600/385100 vertices). We expect that our trained model fails in properly learning the anomalies, thus having a noticeably greater representation error for such nodes. Following this logic, we can see the error of the model as an anomaly score and directly use it for evaluation purposes.

Figure 1 depicts a DualGCN architecture, with 4 input signals forwarded through the twin autoencoders. We choose a factor of $\frac{1}{2}$ to decrease and increase the dimensionality in the encoder and decoder, respectively. Considering this, the architecture from Figure 1, with an input $x \in \mathbb{R}^{N\times 4}$, can at most produce a latent representation space of dimension $N \times 1$. As can be noticed, each of the twin networks yields a representation of the input graph signals, i.e. $\hat{x}_{sym} \in \mathbb{R}^{N\times 4}$ and $\hat{x}_{skew} \in \mathbb{R}^{N\times 4}$. We concatenate these intermediate outputs and the result $\hat{x}_c \in \mathbb{R}^{N\times 8}$ is a representation with a dimension double than the input. Then, the final representation is obtained by means of a linear layer, this way assuring an output $\hat{x} \in \mathbb{R}^{N\times 4}$, identical in size to the input x and so enabling the possibility to directly compute a representation error to further serve as anomaly score.

4. RESULTS

To evaluate the presented method on the aforementioned anomaly detection problem, we have performed 10 individual rounds, *i.e.*, 10 unsupervised training sessions on the *Libra* graph, each followed by an evaluation. We have employed 4 graph signals: number of transactions incoming/outgoing (Nt I/O), total amount incoming/outgoing (Ta I/O). The first two are available directly in the dataset and the latter ones are computed with ease. Following [8], we have chosen $K = 1$ the order of the Chebyshev polynomial. The learning rate used throughout the experiments was 0.1 and the loss function used to determine the representation error was $L1$.

With an identical setup, we evaluate autoencoder models based on $\mathcal{A}_{sym}$ alone, identified as SymGCN, and based on the magnetic Laplacian [4] real and imaginary components, denoted by MagNet. Table 1 shows the details of the evaluated neural models. By 'log' transformation we mean that the natural logarithm is applied to Nt I/O and Ta I/O as input graph signals, or to Ta I/O as weight in the Laplacian.

Table 1. Details of the evaluated neural network models

Model	Input signals	Signals transf.	Weights transf.	Latent dim.
1	Nt I/O	none	none	1
2	Nt I/O	none	log	1
3	Nt I/O	log	none	1
4	Nt I/O	log	log	1
5	Ta I/O	none	none	1
6	Ta I/O	none	log	1
7	Ta I/O	log	none	1
8	Ta I/O	log	log	1
9	Nt I/O, Ta I/O	none	none	2
10	Nt I/O, Ta I/O	none	log	2
11	Nt I/O, Ta I/O	log	none	2
12	Nt I/O, Ta I/O	log	log	2
13	Nt I/O, Ta I/O	none	none	1
14	Nt I/O, Ta I/O	none	log	1
15	Nt I/O, Ta I/O	log	none	1
16	Nt I/O, Ta I/O	log	log	1

We further evaluate the detection capability of our approach against several classical algorithms (non neural network), each computed similarly, in 10 rounds: CALLI [17] – anomaly score embedding the weights and directions of the edges, together with the community structure of the graph; EGO [12] – anomaly detection technique leveraging engineered features of the graph and Isolation Forest [18]; GAW [19] – statistical anomaly score resulting from geometric average of weights and the standardised node degrees. Since CALLI and GAW are directly affected by the weights of the edges, we perform experiments both with logarithmic transformation and without, same as for models in Table 1.

To keep the results concise and comprehensible, we average the detection metrics over the 10 individual rounds of each method. We report the performance for the first 0.1%, 0.2%, 0.5%, and 1% of the most anomalous detected nodes, taking the anomaly weights (number of suspicious transactions in which a node is involved) into account. Moreover, the report also includes the unweighted detection results, which simply mean the number of anomalous nodes. Summarizing, the structure of the table containing the report is: *tpr*_p% – true positives rate for the first p% weighted most anomalous nodes detected; *an*_p% – number of true anomalous nodes among p% unweighted most anomalous nodes detected; *auc*_1% – area under the *tpr* curve for the 1% weighted nodes with highest anomaly scores.

For brevity, out of the 4 models described in each of the 4 families of input graph signals from Table 1, we display the

Table 2. Libra graph results

Method	*tpr*_0.1%	*tpr*_0.2%	*tpr*_0.5%	*tpr*_1%	*auc*_1%	*an*_0.1%	*an*_0.2%	*an*_0.5%	*an*_1%
DualGCN 3	**0.0725**	**0.1218**	**0.2417**	**0.3517**	**0.2186**	**23.00**	**48.00**	**104.50**	**162.10**
SymGCN 3	0.0566	0.1143	0.2221	0.3304	0.2031	17.00	45.20	98.50	152.10
MagNet 3	0.0000	0.0003	0.0021	0.0065	0.0027	0.00	0.30	1.70	4.50
DualGCN 4	**0.0725**	**0.1217**	**0.2419**	**0.3512**	**0.2186**	**23.00**	**47.90**	**104.70**	**162.20**
SymGCN 4	0.0558	0.1066	0.2149	0.3176	0.1951	17.50	40.50	91.70	145.50
MagNet 4	0.0463	0.0875	0.1895	0.2921	0.1731	14.40	32.30	80.60	132.00
DualGCN 5	**0.3086**	**0.4578**	**0.6762**	**0.8259**	**0.6135**	**115.50**	**199.20**	**337.10**	**440.00**
SymGCN 5	0.2669	0.4033	0.6162	0.7725	0.5589	103.00	178.50	307.30	413.90
MagNet 5	0.0016	0.0023	0.0048	0.0069	0.0043	1.40	2.00	4.20	6.00
DualGCN 6	**0.3074**	**0.4588**	**0.6758**	**0.8276**	**0.6130**	**114.80**	**198.80**	**336.90**	**441.90**
SymGCN 6	0.2846	0.4299	0.6416	0.7940	0.5828	108.60	190.00	320.60	423.90
MagNet 6	0.3018	0.4351	0.6510	0.7971	0.5909	110.60	186.20	321.30	420.90
DualGCN 9	**0.3088**	**0.4576**	**0.6760**	**0.8259**	**0.6135**	**115.70**	**199.10**	**337.00**	**440.00**
SymGCN 9	0.1976	0.2953	0.4606	0.6220	0.4270	77.00	128.10	220.20	315.20
MagNet 9	0.0021	0.0028	0.0044	0.0073	0.0045	1.80	2.40	3.90	6.30
DualGCN 10	**0.3086**	**0.4580**	**0.6760**	**0.8263**	**0.6134**	**115.50**	**199.30**	**337.00**	**440.40**
SymGCN 10	0.2734	0.4106	0.6189	0.7697	0.5614	106.30	179.30	307.80	414.00
MagNet 10	0.3069	0.4541	0.6719	0.8197	0.6091	114.30	196.80	334.40	435.80
DualGCN 13	**0.3094**	0.4574	**0.6760**	**0.8259**	**0.6138**	**116.00**	199.00	**337.00**	**440.00**
SymGCN 13	**0.3094**	**0.4578**	**0.6760**	**0.8259**	**0.6138**	**116.00**	**199.20**	**337.00**	**440.00**
MagNet 13	0.0003	0.0019	0.0066	0.0131	0.0064	0.20	1.20	4.50	8.80
DualGCN 14	**0.3094**	0.4574	**0.6760**	**0.8259**	**0.6138**	**116.00**	199.00	**337.00**	**440.00**
SymGCN 14	0.3088	**0.4584**	**0.6760**	**0.8259**	**0.6138**	115.70	**199.50**	**337.00**	**440.00**
MagNet 14	0.3055	0.4484	0.6665	0.8187	0.6062	113.40	194.20	330.80	435.20
GAW_{log}	0.1769	0.3512	0.5871	0.7473	0.5247	71.60	152.50	290.60	406.10
GAW	0.1768	0.3588	0.6150	**0.7601**	0.5402	71.40	156.50	303.90	**413.20**
CALLI_{log}	0.0832	0.1412	0.2400	0.3273	0.2173	29.20	56.30	109.60	155.40
CALLI	0.0972	0.1518	0.2399	0.3154	0.2194	35.10	63.70	111.40	153.50
EGO	**0.4004**	**0.5051**	**0.6559**	0.7444	**0.6016**	**160.90**	**225.90**	**317.40**	377.20

results only for the configurations which produced the best two AUC values within each family. Table 2 presents in bold the largest metric value for each model.

The results for models 1–4 are poor, showing that the number of transaction alone is not a relevant input signal for our problem. Employing Ta I/O as input signals in 5–8 with a model having the same latent dimension 1, we can observe that the transferred amounts, although structurally included in the networks within the Laplacians as weights, have the greatest discriminative power in uncovering anomalous nodes. DualGCN has the best detection performance, closely followed by MagNet when the weights transformation is present.

Further combining Nt I/O and Ta I/O in models 9–12 with a latent size 2, there are no decisive improvements for DualGCN and SymGCN. For models 13–16, DualGCN keeps obtaining top results and SymGCN is finally able to match them. Overall, the behavior of DualGCN is quite robust and shows good results for most of the models, while SymGCN obtains its best results only in the most complex configuration (deepest network).

MagNet is constantly outperformed by our networks. Note also that weights transformation plays a crucial for MagNet, the results without log transformation being extremely poor, due to the lack of phase scaling.

When compared to non neural network detection techniques, DualGCN is outperformed only by EGO and only with respect to the first 0.1% and 0.2% of the most anomalous bank accounts. Comparing DualGCN and EGO in terms of *auc*_1%, we can see relatively close values. Oppositely, large discrepancies can be observed in *an*_p%. We can deduce that EGO tends to find nodes with larger anomaly weights, whereas DualGCN uncovers more anomalous nodes. A similar interpretation explains the results of GAW.

Although DualGCN is composed of a pair of twin autoencoder networks, the training operation takes less than for SymGCN. We set an early stopping criterion in order to end the training when the loss over the last 10 epochs varies by less than 1%. Throughout the experiments, the training of DualGCN models consistently finishes in under 20 epochs, roughly translated to less than 3.5 seconds for the most com-

plex DualGCN models (13–16), and less than 1.8 seconds for the simpler ones (1–12), on an Apple MacBook M2 Pro. On the other hand, SymGCN and MagNet are more difficult to train and require a lot more epochs (sometimes more than 100). Their training may take more than 10 seconds on the same machine. For comparison, EGO needs about 30 minutes for computing the 14 features used for anomaly detection, while CALLI and GAW require 5-10 minutes.

5. CONCLUSIONS

In this paper, we have proposed a GCN architecture for directed graphs, based on their symmetrized and skew-symmetrized Laplacian matrices. Then, we tackled the anomaly detection problem in a real graph and compared the results obtained with the proposed DualGCN method against several other approaches. Our method proved to be superior, outperforming both classical and neural network techniques. Moreover, the experiments on the *Libra* graph showed that DualGCN is computationally inexpensive and feasible for real use-cases, thus hinting towards the possibility of targeting anomaly detection problems in applications from a multitude of domains.

6. REFERENCES

[1] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*. PMLR, 2016, pp. 2014–2023.

[2] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun, "Spectral networks and deep locally connected networks on graphs," in *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

[4] X. Zhang, Y. He, N. Brugnone, M. Perlmutter, and M. Hirn, "Magnet: A neural network for directed graphs," *Advances in neural information processing systems*, vol. 34, pp. 27003–27015, 2021.

[5] Y. He, M. Perlmutter, G. Reinert, and M. Cucuringu, "Msgnn: A spectral graph neural network based on a novel magnetic signed laplacian," in *Learning on Graphs Conference*. PMLR, 2022, pp. 40–1.

[6] J. Park, M. Lee, H.J. Chang, K. Lee, and J.Y. Choi, "Symmetric graph convolutional autoencoder for unsupervised graph representation learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6519–6528.

[7] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, 2016.

[8] T.N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *CoRR*, vol. abs/1609.02907, 2016.

[9] Y. Ma, J. Hao, Y. Yang, H. Li, J. Jin, and G. Chen, "Spectral-based graph convolutional network for directed graphs," *CoRR*, vol. abs/1907.08990, 2019.

[10] Z. Tong, Y. Liang, C. Sun, D.S. Rosenblum, and A. Lim, "Directed graph convolutional network," *CoRR*, vol. abs/2004.13970, 2020.

[11] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P.S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[12] B. Dumitrescu, A. Băltoiu, and Ş. Budulan, "Anomaly detection in graphs of bank transactions for anti money laundering applications," *IEEE Access*, vol. 10, pp. 47699–47714, 2022.

[13] F.R.K. Chung, *Spectral graph theory*, vol. 92, American Mathematical Soc., 1997.

[14] M. Lampert and I. Scholtes, "The self-loop paradox: Investigating the impact of self-loops on graph neural networks," *CoRR*, vol. abs/2312.01721, 2023.

[15] D.K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, Mar. 2011.

[16] S. Gerschgorin, "Uber die abgrenzung der eigenwerte einer matrix," *Izvestija Akademii Nauk SSSR, Serija Matematika*, vol. 7, no. 3, pp. 749–754, 1931.

[17] T.–A. Badea and B. Dumitrescu, "Community-augmented local-link intensity: A score for anomaly detection in graphs," in *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2023, pp. 1936–1941.

[18] F.T. Liu, K.M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, USA, 2008, ICDM '08, p. 413–422, IEEE Computer Society.

[19] A. Elliott, M. Cucuringu, M.M. Luaces, P. Reidy, and G. Reinert, "Anomaly detection in networks with application to financial transaction networks," *ArXiv*, vol. abs/1901.00402, 2019.