i-DGCN: A Spectral Convolutional Network For Directed Graphs Using An Intensity Laplacian

Theodor-Adrian Badea, Bogdan Dumitrescu
Department of Automatic Control and Computers
National University of Science and Technology Politehnica Bucharest
Bucharest, Romania
theodor.badea@upb.ro, bogdan.dumitrescu@upb.ro

***Abstract*—The development of graph neural networks has been driven by the widespread area of applications where graphs are naturally fit and by the advances in making solutions scalable. When it comes to spectral graph convolutional networks (GCNs), directed graphs suffer from the asymmetric nature of their Laplacian matrices. For such graphs, there is no natural extension of the spectral graph theory well-established for their undirected analogues with inherent symmetric matrices. In this paper, we propose *i*-DGCN: a spectral GCN approach addressing directed graphs by means of a novel symmetric Laplacian matrix constructed using a quantification of the interaction between the nodes. In order to assess *i*-DGCN, we undertake two tasks: anomaly detection for graph-structured data (unsupervised) and graph link existence prediction (supervised). Then, we compare the results with other Laplacian alternatives for directed graphs.**

***Index Terms*—graph neural networks, spectral graph convolution, directed graphs**

I. Introduction

Graph machine learning is an effervescent field that extends the traditional machine learning methods – often assuming data points are independent, to graphs – where data is inherently interdependent. An appealing part of this realm is represented by spectral graph convolutional networks (GCNs), which derive the principles of convolutional neural networks to graph data. GCNs operate in the spectral domain, using the eigenvalues and eigenvectors of the graph Laplacian to perform convolution operations. This approach allows GCNs to generalize the concept of spatial locality, which is essential to the irregular structure of graphs.

Applying spectral methods to directed graphs presents unique challenges. The graph Laplacian, a fundamental component in spectral methods, does not enjoy the neat properties of a symmetric matrix. This generally translates to complex eigenvalues and the lack of well-defined basis for convolution.

A. Content and contribution

In this paper, we propose an artificial Laplacian matrix for directed graphs, constructed by means of representing the relationships between nodes in a symmetric way. This is achieved through quantifying the interactions between vertices and obtaining an *intensity* [1]. The suggested Laplacian is based on the adjacency matrix built with intensities rather than edge weights. We then address two graph problems: anomaly detection and link existence prediction.

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.

B. Related work

For undirected graphs, a significant leap in defining spectral GCNs was taken in [2], where an architecture based on Chebyshev polynomials of the adjacency matrix was used. For directed graphs, such a solution can be immediately used if we work with the symmetrized adjacency matrix (instead of the true asymmetric Laplacian). This approach can work in a limited number of applications, since it ignores the imbalances between nodes.

To include asymmetry, the key is to recur to Hermitian Laplacians, where the information introduced by the imaginary part—or the phase—can model the direction of the edges. MagNet [3] was the first to propose the use of the magnetic Laplacian where, speaking in broad terms, the symmetrized adjacency matrix takes the role of magnitude and the prevalent direction of the edges gives the phase. MagNet needs careful scaling of the phase, because its periodic nature can make the related information ambiguous. An alternative, working also for signed weights, is the sign-magnetic Laplacian (SigMaNet) [4], where the directions of the edges and the signs of the weights are uniquely coded by the phase. Other Laplacians from the same family can be found in [5]–[7]. An extension to quaternions was proposed in [8].

Some previous substitutes of the Laplacian for directed graphs used several symmetric matrices [9] and thus were more complicated than a Hermitian Laplacian.

II. Spectral graph convolutions

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph, with $|\mathcal{V}| = N$ nodes. $\mathcal{E}$ is composed of unordered pairs of vertices from $\mathcal{V}$ and let us denote $w_{uv} > 0$ the weight of the edge between vertices u and v, leading to the construction of the adjacency matrix $\mathcal{A}$ and of the diagonal degree matrix $\mathcal{D}$, with $\mathcal{D}(u,u) = \sum_{v \neq u} w_{uv}$.

Graphs lack a relevant translation operator, therefore the spatial convolution is impossible to define. However, the graph Fourier domain enables convolutions through multiplication. The foundation of defining the Fourier transform (and its inverse) for a graph signal is the eigendecomposition of the Laplacian $\mathcal{L} = \mathcal{D} - \mathcal{A}$, which can be diagonalized as $\mathcal{L} = U \Lambda U^T$, where $U = [u_0, ..., u_{N-1}]$ is the matrix of eigenvectors (Fourier basis) and $\Lambda = \text{diag}([\lambda_0, ..., \lambda_{N-1}])$ is the diagonal matrix of eigenvalues (frequencies). The Laplacian of an undirected graph is a symmetric matrix, therefore enjoys a complete set of orthonormal eigenvectors together with real and nonnegative eigenvalues. In practice, the classic combinatorial Laplacian $\mathcal{L} = \mathcal{D} - \mathcal{A}$ is often substituted by the symmetrically normalized form [10]:

$$\mathcal{L}_s = \mathcal{D}^{-1/2} \mathcal{L} \mathcal{D}^{-1/2} = \mathcal{I}_N - \mathcal{D}^{-1/2} \mathcal{A} \mathcal{D}^{-1/2}, \qquad (1)$$

with eigenvalues in $[0, 2]$.

With a proper convolution defined, a signal x can be filtered by a filter g_θ according to [11]:

$$g_\theta *_\mathcal{G} x = g_\theta(\mathcal{L}_s) x = g_\theta\left(U \Lambda U^T\right) x = U g_\theta(\Lambda) U^T x. \quad (2)$$

The simplest g_θ can be a non-parametric filter, however, as emphasized in [11], such a filter is not localized. Moreover, [2] pinpoints as drawbacks the computational complexity of performing the convolution and the difficulty to scale the approach for large graphs because of the eigendecomposition.

A parametrization of the filter with a recursively-formulated polynomial evaluated in the Laplacian can be employed to tackle these issues, as [11] suggests. It is further shown in [12] that a truncated expansion in terms of Chebyshev polynomials up to order K can be used to approximate the filter $g_\theta(\Lambda)$ and so the filtering operation becomes:

$$g_{\theta'} *_\mathcal{G} x = \sum_{k=0}^{K} \theta'_k T_k(\tilde{\mathcal{L}}_s) x, \qquad (3)$$

where θ'_k are coefficients, $T_k(t) = 2t T_{k-1}(t) - T_{k-2}(t)$ with $T_0 = 1$ and $T_1 = t$, and $\tilde{\mathcal{L}}_s = \frac{2}{\lambda_{max}} \mathcal{L}_s - \mathcal{I}_N$. Following [2], $\lambda_{max} = 2$ and $K = 1$ can be employed. Moreover, a *renormalization* is suggested, consisting in initially adding self-loops to the adjacency matrix and then proceeding to computing the Laplacian $\tilde{\mathcal{L}}_s$ starting from $\tilde{\mathcal{A}} = \mathcal{A} + \mathcal{I}_N$. This renormalization procedure is motivated by making the gradients more stable when iteratively applying the Laplacian operator, but also by aggregating information from itself in the convolution, not only from neighboring vertices.

III. *i*-DGCN

Instead of attempting to generalize the spectral graph theory to the family of directed graphs, a more reasonable approach would be to derive a symmetric Laplacian alternative.

A. *Intensity*

In [1], the following intensity is introduced, as a means of quantifying the interaction between vertices:

$$i_v(u) = \begin{cases} \max\left(\mathcal{P}_{gaw}(u), \; \mathcal{S}_{gaw}(v)\right), & \text{if } v \in \mathcal{P}(u) \\ \max\left(\mathcal{P}_{gaw}(v), \; \mathcal{S}_{gaw}(u)\right), & \text{if } v \in \mathcal{S}(u), \end{cases} \qquad (4)$$

where $\mathcal{P}(u)$ and $\mathcal{S}(u)$ denote the sets of predecessors and successors of u, respectively. Similarly, $\mathcal{P}_{gaw}(u)$ and $\mathcal{S}_{gaw}(u)$ are the geometric averages of weights associated with edges between u and $\mathcal{P}(u)$ and $\mathcal{S}(u)$, respectively. This is a measure of the strength of the interaction of node u w.r.t. node v, constructed considering both edge weights and directions.

We can leverage the intensity and derive a symmetric adjacency matrix of $\mathcal{G}$. In other words, we indirectly alter the structure of the graph by replacing a directed and weighted edge with an undirected edge which has the intensity assigned as weight. The benefit of this approach is that it captures the importance of a node in the context of its neighbors. Since the intensity measures the interaction of node u w.r.t. node v, we can notice the following equality holds when vertices u and v are connected by a single edge:

$$i_v(u) = i_u(v). \qquad (5)$$

For simplicity, we further denote by i_{uv} the unique value characterizing the connection between u and v, regardless of the original edge direction.

However, the case where two edges of opposing directions exist between the same nodes needs to be accommodated too. We handle this situation by considering the maximal intensity between the two vertices.

B. *Intensity Laplacian*

Using the intensity measure, we can construct a corresponding adjacency matrix, defined as:

$$\mathcal{A}_i(u, v) = \begin{cases} i_{uv}, & \text{if } (u, v) \in \mathcal{E} \\ 0, & \text{otherwise.} \end{cases} \qquad (6)$$

Since $\mathcal{A}_i$ is symmetric (moreover, positive semidefinite), it is well-behaved w.r.t. the notions presented in Section II and can, therefore, be used in deriving the intensity Laplacian

$$\mathcal{L}_{is} = \mathcal{I}_N - \mathcal{D}_i^{-1/2} \mathcal{A}_i \mathcal{D}_i^{-1/2}. \qquad (7)$$

This way, our Laplacian offers not only a well-behaved operator from a spectral standpoint, but also a glimpse of the most dominant vertices in the graph and embeds a quantification of the interaction between nodes which inherently takes into account original edge directions.

C. i-*DGCN architecture*

By *i*-DGCN we denote the approach of using spectral convolutional layers based on the proposed intensity Laplacian rather than denoting a specific architecture. Let L be the number of layers and let $X^{(0)}$ be a $N \times S_0$ matrix of input features, where each column is a graph signal $x_k^{(0)}$, $k \in \{1, 2, ..., S_0\}$. Considering (3) and approximating $\lambda_{max} = 2$, we can write the propagation rule through a layer as

$$X^{(l+1)} = \sigma\left(\sum_{k=0}^{K} T_k(\tilde{\mathcal{L}}_{is}) X^{(l)} \Theta^{(l)}\right), \qquad (8)$$

where $\tilde{\mathcal{L}}_{is}$ denotes the symmetrically normalized intensity Laplacian, with the renormalization considered. $\Theta^{(l)}$ represents the learnable filter parameters of the layer, and σ the nonlinearity. After propagation through layer l, the resulting $X^{(l+1)}$ is a matrix $N \times S_{l+1}$, where S_{l+1} depends on the characteristics of the network.

IV. Results

As already mentioned, we assess the proposed Laplacian by tackling the anomaly detection (unsupervised) and link existence prediction (supervised) problems, for which we run experiments[1] with various configurations.

Besides *i*-DGCN, we train similar models based on the Magnetic Laplacian (MagNet) [3], Sign-Magnetic Laplacian (SigMaNet) [4], and the trivially symmetrized Laplacian, denoted SymGCN (the weights of edges (u, v) and (v, u) are equal to $(w_{uv}+w_{vu})/2$). For both tasks, the employed learning rate values are 0.001, 0.005, 0.01, and 0.05. The order of the Chebyshev polynomial is 1. We use Adam optimizer and, to avoid overfitting, *l*-2 regularization with hyperparameter set to $5 \cdot 10^{-4}$. Moreover, a final dropout layer of probability 0.5 is present for each model in the link prediction task and after each convolutional layer for anomaly detection.

1) Anomaly detection: For the first task, we integrate our Laplacian into the spectral convolutional layers of a symmetric autoencoder network [13] and attempt to detect anomalies in a directed graph, in an unsupervised manner. The dataset is adopted from [14], a graph composed of transactions between bank accounts belonging to customers of *Libra Internet Bank* [14], with 385100 vertices and 597165 weighted edges. Anomalies represent accounts suspected of money laundering and sum up to a total of 600 individual nodes, with a total weight of 1034 (each node has an anomaly weight proportional with the number of anomalous transactions it is involved in). We feed graph signals through each convolutional layer of the network, constructed as shown in (8). The classification procedure itself relies on the obvious class imbalance between anomalous and normal nodes of the *Libra* graph, expecting the network to fail in properly learning a latent representation and then reconstructing anomalous nodes. Hence the representation error can be seen as a direct anomaly score.

We set the size of the encoder input equal to the size of the decoder output and each propagation through the encoder and decoder is characterized by decreasing and, respectively, increasing the dimensionality of the data by a factor of 2. Noticeably, there can be a maximum number of layers limited by the chosen size of the encoder input and decoder output. Moreover, in case of dimension discrepancies, we handle them by making use of two linear layers which surround the encoder-decoder structure and properly adapt the dimension of the data. We use the *l*-1 loss function, to allow a few large errors, presumably for anomalies, and ReLU [15] as nonlinearity. The choice of graph signals for these experiments are the incoming and outgoing amounts for a vertex, therefore an input set of size $N \times 2$.

Our experiments use two autoencoder flavors, depending on the hidden dimension: **complete** and **overcomplete**. The first architecture has a hidden dimension equal to the number of input signals, i.e. 2, whereas for the second one, we choose 4 as hidden dimension. In order to eliminate any possibility of encountering phase scaling issues [4] for the magnetic Laplacian, we take the logarithm of the original weights.

[1]Code available at https://github.com/theodorbadea/i-DGCN

TABLE I
Anomaly detection TPR-AUC results for Libra dataset

Model	*auc*_1%	*tpr*_0.1%	*tpr*_0.2%	*tpr*_0.5%	*tpr*_1%
		COMPLETE			
***i*-DGCN**	**61.442**	30.812	**46.411**	**67.678**	**82.688**
MagNet	61.371	30.909	45.831	67.611	82.620
SigMaNet	61.380	**30.947**	45.744	67.601	82.591
SymGCN	61.266	30.802	45.793	67.524	82.533
		OVERCOMPLETE			
***i*-DGCN**	**61.488**	30.889	**46.566**	**67.698**	**82.649**
MagNet	61.384	**30.947**	45.744	67.601	82.591
SigMaNet	61.384	**30.947**	45.744	67.601	82.591
SymGCN	61.383	**30.947**	45.744	67.601	82.591

TABLE II
Anomaly detection AN results for Libra dataset

Model	*an*_0.1%	*an*_0.2%	*an*_0.5%	*an*_1%
		COMPLETE		
***i*-DGCN**	115.3	**202.1**	**337.5**	**442.0**
MagNet	115.8	199.3	337.1	440.3
SigMaNet	**116.0**	199.0	337.0	440.0
SymGCN	114.9	198.8	336.4	440.0
		OVERCOMPLETE		
***i*-DGCN**	115.7	**202.9**	**338.0**	**441.4**
MagNet	**116.0**	199.0	337.0	440.0
SigMaNet	**116.0**	199.0	337.0	440.0
SymGCN	**116.0**	199.0	337.0	440.0

The *Libra* graph is prone to training overfitted autoencoder models, thus we enforce maximum number of 100 epochs, with a rather aggressive early stopping condition of 10 epochs without finding a smaller loss. We performed 10 independent training and evaluation sessions with each model. Since there are only $600/385100$ anomalies in the Libra graph, we report the results such that we alleviate the obvious class imbalance: *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. We choose p as 0.1, 0.2, 0.5, and 1. Furthermore, we show these values as percentages, where 100% corresponds to 1.

Tables I and II show the best performing model of each type in terms of *auc*_1% averaged across the 10 individual rounds. For both complete and overcomplete autoencoder variants, the best model of each family is yielded by a 0.05 learning rate – the largest used throughout our experiments.

Our intensity Laplacian is clearly capable, yielding models which outperform the magnetic Laplacians and the trivially symmetrized alternative on the anomaly detection task. *i*-DGCN wins in terms of *auc*_1% and TPR, with a single exception for *tpr*_0.1%, where SigMaNet obtains the largest value and *i*-DGCN is ranked third. These results are perfectly consistent with the number of actual anomalous nodes uncovered, as shown in Table II. For *an*_0.2% and *an*_1%, *i*-DGCN manages to uncover approximately two more anomalous nodes than the second best, MagNet. The overcomplete architecture has a dimension of 4, hence two layers are needed for the encoder to reach a latent representation space of size 1 and the same number of layers for the decoder to produce a reconstruction of the same original dimension. Also, the additional linear layers are employed as described to transform the data into the appropriate shape. As we can see in Tables I and II, the same perfectly consistent behavior as in the complete autoencoder case occurs: *i*-DGCN loses only for *tpr*_0.1% and *an*_0.1%. Adding more complexity to the network slightly improved the results for our approach and partially for others. MagNet does not benefit from the overcomplete architecture, results being better only for *tpr*_0.1% and poorer for all the other metrics when put side by side with its complete variant. SymGCN equals the magnetic approaches in this setup.

Although our intensity Laplacian slightly outperforms other alternatives, it is fair to say that they all perform relatively well. The results are clearly better than those of the reduced egonet method from [14], where the values for *auc*_1% and *an*_1% are 60.16 and 377.2, respectively; however, the cited method is still the best for *tpr*_0.1%, with a result of 40.04.

2) Link existence prediction: To predict the existence of edges, we make use of convolutional architectures with various dimensions, as in [3], [4]. Chebyshev convolutional layers are stacked together with ReLU [15] nonlinearity. After the convolutions, the signal can be seen as a node embedding and a final linear transformation is applied. Then, logarithmic softmax is employed and the model is trained through negative logarithmic likelihood loss between the predicted and the actual edge existence label. Three graphs are used for this task: *Telegram* [16]; *Bitcoin Alpha* and *Bitcoin OTC* [17]. The first dataset is a pairwise-influence network between Telegram channels used to analyse interactions related to the propagation of political ideologies. It has 245 nodes belonging to four classes and 8912 edges. Bitcoin graphs originate from the eponymous cryptocurrency exchanges and can be viewed as trust networks: both *Alpha* and *OTC* exchanges allow users to rate the others on a scale of -10 to $+10$ (0 excluded), where scammers should be assigned -10 and fully legitimate users $+10$. Bitcoin Alpha has 3783 vertices with 22650 and 1536 positive and negative edges, respectively. The slightly larger Bitcoin OTC has 5881 nodes with 32029 and 3563 positive and negative edges, respectively. Since our method does not support negative weights, we preprocess these datasets as in [4]. Similarly to anomaly detection, we use the incoming and outgoing total edge weights as graph signals.

Since the approach is supervised, we follow [3], [4] and split the datasets for training (80%), validation (5%), and testing (15%). The experiments are run with k-cross validation, where $k = 10$, and preserving graph connectivity. We search models among variations with 2, 4, 8 layers and a hidden dimension of 16, 32, 64. The maximum number of epochs in the training phase is set to 3000, with a more relaxed early stopping condition of 500 iterations without the validation error decreasing.

Two flavors of this task are addressed, as described in [3]:

- **(a) Existence prediction**: if $(u, v) \in \mathcal{E}$, (u, v) has the label 0, otherwise 1;
- **(b) Three-class link prediction**: for an ordered pair (u, v), the label is 0 if $(u, v) \in \mathcal{E}$; the label is 1 if $(v, u) \in \mathcal{E}$; and the label is 2 if $(u, v) \notin \mathcal{E}$ and $(v, u) \notin \mathcal{E}$.

TABLE III
LINK EXISTENCE PREDICTION.

Model	Telegram	Bitcoin Alpha	Bitcoin OTC
Existence prediction			
***i*-DGCN**	**86.94** ± 0.49	**87.61** ± 0.54	**88.89** ± 0.24
MagNet	86.87 ± 0.57	87.42 ± 0.42	88.61 ± 0.56
SigMaNet	86.38 ± 0.38	87.27 ± 0.44	88.34 ± 0.46
SymGCN	86.44 ± 0.61	87.17 ± 0.37	88.64 ± 0.31
Three-class link prediction			
***i*-DGCN**	83.03 ± 0.50	**85.32** ± 0.54	85.47 ± 0.73
MagNet	**83.12** ± 0.42	85.29 ± 0.67	**85.53** ± 0.67
SigMaNet	82.40 ± 0.36	84.87 ± 0.51	85.51 ± 0.85
SymGCN	82.66 ± 0.52	85.25 ± 0.73	85.43 ± 0.81

In [4], we can notice that MagNet and SigMaNet prove to outperform other approaches when tackling the link existence prediction problem. Therefore, we compare our Laplacian only with these two magnetic alternatives. Additionally, we also test using the symmetrized Laplacian.

Table III contains the results obtained, averaged across the 10 splits, together with the standard deviation. The results show that our Laplacian achieves the best results on all three datasets for the binary existence task, outperforming both magnetic and the trivially symmetrized Laplacians. For *i*-DGCN and SymGCN, the best model for each dataset has a configuration with 8 layers and 64 hidden dimension. Slightly differently, MagNet attains best results with only 4 layers for Bitcoin Alpha and Bitcoin OTC. As for the learning rates, we could notice that *i*-DGCN and SymGCN obtain the best results 0.001 and 0.005, whereas magnetic Laplacians need larger learning rates.

On the three-class link prediction task, even though the intensity Laplacian does not explicitly encode edge orientations, it still manages to outperform the other alternatives on the Bitcoin Alpha dataset. Similarly, for all three graphs, the best model yielded by the *i*-DGCN approach had 64 hidden dimension and 4 and 8 layers for the two Bitcoin and Telegram graphs, respectively.

V. Conclusions

We have proposed a method to construct a Laplacian matrix for directed and weighted graphs starting from an intensity measure [1], which characterizes the interaction between vertices in a symmetric manner. The advantage is that our Laplacian, by construction, inherently embeds information about the vertex dominance within the graph w.r.t. the original orientations of the edges. Then, we have integrated it in the layers of spectral GCNs, denoted by *i*-DGCN. We have evaluated the proposed Laplacian for both unsupervised and supervised tasks: anomaly detection and two flavors of link existence prediction, respectively. In each scenario, *i*-DGCN proved either clearly superior, or a serious competitor to other Laplacian alternatives for directed and weighted graphs.

REFERENCES

[1] Theodor–Adrian Badea and Bogdan 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)*, pages 1936–1941, 2023.

[2] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017. preprint arXiv:1609.02907.

[3] Xitong Zhang, Yixuan He, Nathan Brugnone, Michael Perlmutter, and Matthew Hirn. Magnet: A neural network for directed graphs. *Advances in neural information processing systems*, 34:27003–27015, 2021.

[4] S. Fiorini, S. Coniglio, M. Ciavotta, and E. Messina. SigMaNet: One Laplacian to rule them all. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7568–7576, 2023.

[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*, pages 40–1. PMLR, 2022.

[6] R. Singh and Y. Chen. Signed graph neural networks: A frequency perspective. *arXiv preprint arXiv:2208.07323*, 2022.

[7] T. Ko, Y. Choi, and C.K. Kim. A spectral graph convolution for signed directed graphs via magnetic laplacian. *Neural Networks*, 164:562–574, 2023.

[8] S. Fiorini, S. Coniglio, M. Ciavotta, and E. Messina. Graph Learning in 4D: A Quaternion-Valued Laplacian to Enhance Spectral GCNs. In *Proc. AAAI Conference on Artificial Intelligence*, volume 38, pages 12006–12015, 2024.

[9] Zekun Tong, Yuxuan Liang, Changsheng Sun, David S Rosenblum, and Andrew Lim. Directed graph convolutional network. *arXiv preprint arXiv:2004.13970*, 2020.

[10] F.R.K. Chung. *Spectral Graph Theory*. Number nr. 92 in CBMS Regional Conference Series. Conference Board of the Mathematical Sciences.

[11] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering, 2017.

[12] David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, March 2011.

[13] Jiwoong Park, Minsik Lee, Hyung Jin Chang, Kyuewang Lee, and Jin Young Choi. Symmetric graph convolutional autoencoder for unsupervised graph representation learning, 2019.

[14] Bogdan Dumitrescu, Andra Băltoiu, and Ştefania Budulan. Anomaly detection in graphs of bank transactions for anti money laundering applications. *IEEE Access*, 10:47699–47714, 2022.

[15] Kunihiko Fukushima. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333, 1969.

[16] Alexandre Bovet and Peter Grindrod. The activity of the far right on telegram v2.11, 12 2020.

[17] Srijan Kumar, Francesca Spezzano, V. S. Subrahmanian, and Christos Faloutsos. Edge weight prediction in weighted signed networks. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 221–230, 2016.