Graph-Based Neural Networks for Anti-money Laundering: Experimenting with Dense Graph Convolutional Networks for Bitcoin Transactions

1st Vasileios Moschopoulos
MSc Data and Web Science
Aristotle University of Thessaloniki
Thessaloniki, Greece
will.moschopoulos@gmail.com

2nd Georgios Michoulis
MSc Data and Web Science
Aristotle University of Thessaloniki
Thessaloniki, Greece
gmixoulis@gmail.com

***Abstract*—In this paper we study the fraud detection field in the domain of Bitcoin transactions, using the Bitcoin Elliptic dataset. Regarding fraud detection, graph-based anomaly detection combined with machine learning has been heavily used in recent years in order to detect fraudulent users in networks. Through the graph representation of networks, machine learning models are able to interpret the interdependencies between the objects of a network, leading to more efficient identification of anomalies, including fraudulent users, as in our case. We also compare these approaches with a Graph Convolutional Network (GCN) classifier.**

***Index Terms*—Graphs, Network analysis, Anomaly detection, Fraud detection, Graph Convolutional Networks, Bitcoin**

I. Introduction

Modern, widespread digitalization has given rise to the creation of global networks. People can now easily access and interact with such networks, although, sometimes, with malicious intent. Such global networks include social networks (Facebook, Twitter, Instagram, etc.), banking networks, websites, cryptocurrency networks, and telecommunication networks, where people can access large amounts of daily produced data for their benefit. Social networks have been used to influence election outcomes in recent years due to their impact. Another example, regarding banking networks, is people trying to take advantage of their access to the network in order to intercept people's bank accounts or transactions. This type of user's malicious behavior led to the fraud detection field, which detects fraudulent users in a network. In this work, we study Bitcoin fraud detection using the Bitcoin Elliptic dataset. For that reason, we trained several models and compared their results to reach a conclusion about our goal.

Graph-based anomaly detection combined with machine learning is used to detect fraudulent network users. Through graph representation of networks, machine learning models can interpret interdependencies between network objects, leading to more efficient identification of anomalies, including fraudulent users. In this framework, we compare Random Forest, Multi-Layer Perceptron, and logistic regression classifier performance. We compare these approaches to a Graph Convolutional Network (GCN) classifier, a modern technique, for which other works [1]–[3] have presented a slightly different approach. Compared to the two-layer GCN implementation in [3], which introduced the Elliptic Bitcoin dataset, we propose a three-layer GCN architecture with a linear output layer.

In this study, we present the results of our research regarding the field of fraudulent user detection in the field of Bitcoin transactions, through the various trained models and the comparison among them. Specifically, our contributions are:

- A data preprocessing procedure and intuitive graph visualizations of the Elliptic dataset utilizing the NetworkX library and Gephi platform.
- A framework for training baseline machine learning algorithms, such as logistic regression, random forest classifier, and multi-layer perceptron classifier, evaluating them, and using random forest classifier to classify the unknown class of the dataset and its graph visualization.
- The employment of a more sophisticated approach, i.e., a three-layer GCN architecture as an enhancement of the study presented in [3] including the corresponding data preprocessing procedure and the corresponding training and evaluation.

II. Related Works

This section contains two types of related work. Firstly, algorithmic fraud detection approaches, use shallow learning to detect malicious behavior. Usama Fayyad [4] developed an adaptive model that checks user behavior using data mining techniques. The REV2 [5] algorithm is introduced regarding fraudulent user prediction in rating platforms, proposing three interdependent intrinsic quality metrics and axioms to establish their interdependency: user fairness, rating reliability, and product goodness. The Fraudar algorithm [6] takes 'camouflaged' users in bipartite graphs related to online product reviews into account to detect fake reviews in real-world data. The second category of related work algorithms use GCN-based models. Sushmito Ghosh [7] first presents credit card fraud detection neural network results. Kaize Ding and his colleagues [1] propose a GCN-based auto-encoder neural network that recreates the initial graph and classifies anomalous nodes based on residual errors. Then, a case of insider thread and fraud detection [2] using a 2-layer GCN is presented to detect anomalous user behavior and malicious threat groups. Li Zheng et al. [8] proposed AddGraph for detecting anomalies in dynamic graphs by combining a GCN and an attention-based GRU.

III. Network Analysis

In this section, we describe our preprocessing work on the Bitcoin transaction (Elliptic) dataset and present the graphs plots used to visualize the dataset.

For our data, we utilized the Elliptic dataset which is a sub-graph of the bitcoin graph, made of 203,769 nodes and 234,355 edges. The release of the current dataset is attended by a new tutorial paper that Elliptic data scientists have co-authored with researchers from the MIT-IBM Watson AI Lab [3]. It categorizes the nodes into three classes namely *licit*, *illicit* and *unknown*. A node is deemed licit if the corresponding transaction that has been created by a user belongs to a licit category i.e., exchanges, wallet providers, miners, and financial service providers. On the hand, a node is considered illicit when a transaction belongs to an illicit category i.e., scams, malware, terrorist organizations, ransomware, and Ponzi schemes. Finally, the unknown category that accounts for the majority of the observations in comparison with the other two aforementioned categories is formed by transactions that have not been classified yet.

This dataset includes a range of time steps ranging from 1 to 49 as a measure of the actual transaction time stamp. The time steps are evenly spaced with an interval of about two weeks. Each time step contains a single connected component of transactions that appeared on the blockchain within less than three hours between each other. The timesteps are not connected with each other with edges. Considering the fact that each time step represents 2 weeks, the overall time period captured by the dataset amounts to 98 weeks.

The dataset was used in our framework to accurately predict fraud and non-fraud transactions using a large amount of licit and illicit data. Additionally, a goal was to classify the unknown category as licit or illicit through the given timesteps and visualize preprocessing and predicted graphs with the NetworkX library and Gephi platform.

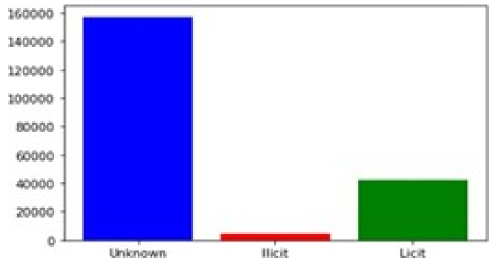

Fig. 1. Comparison of number of transactions in the dataset

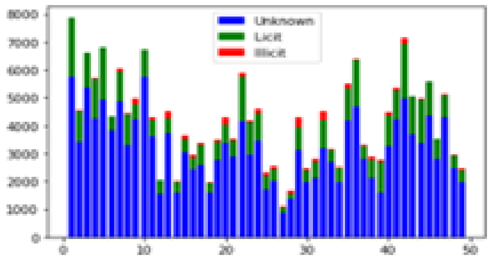

Fig. 2. Total number of transactions in each time step

IV. Data preprocessing for the utilized baseline algorithms

This section explains the data preprocessing steps. Classes, features, and edge lists comprised the data. Figure 1 shows that the majority of transactions are unknown and the minority are illicit. Figure 2 shows the transaction distribution over time.

In the illicit case, several connected components have the same number of nodes and edges, so the giant component is small. In the licit case, the overall graph density is higher. Hubs with high centrality and betweenness are trusted and perform many transactions quickly. Peripheral users complete transactions with a limited circle of users, such as relatives or trusted coworkers.

Gephi helped us create accurate graphs and evaluate our dataset. Figure 2 shows that the volume of transactions varies by timestep, and, thus, we decided to evaluate based on the maximum timestep, 43. Gephi helps to visualize graph-based datasets and mines valuable knowledge. To visualize our time step, we used Gephi's:

- ForceAtlas2. This algorithm is a force-directed layout. It simulates a physical system in order to spatialize a network. Nodes repulse each other like charged particles, while edges attract their nodes, like springs. These forces create a movement that converges into a balanced state. This final configuration is expected to help the interpretation of the data [9].
- Fruchterman Reingold. This algorithm is a force-directed layout algorithm. The idea of a force-directed layout algorithm is to consider a force between any two nodes. In this algorithm, the nodes are represented by steel rings and the edges are springs between them. The attractive force is analogous to the spring force and the repulsive force is analogous to the electrical force. The basic idea is to minimize the energy of the system by moving the nodes and changing the forces between them [10].

Figure 3 shows Gephi's Reingold result. By coloring the betweenness centrality, we can see that 5 nodes have the highest centrality. For every pair of vertices in a connected graph, there is the shortest path that minimizes the number of edges. Betweenness centrality is the number of shortest paths through a vertex. Five nodes control many transactions in our transaction-focused dataset. ForceAtlas2 algorithm can show high betweenness centrality.

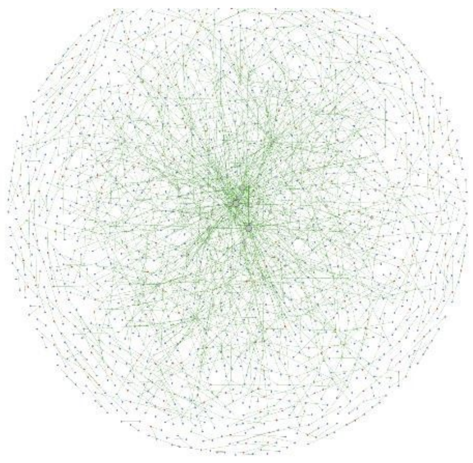

Fig. 3. Licit user to user transaction in the 43rd timestamp using Reingold.

Fig. 4. Licit user to user transactions in the 43rd timestamp using ForceAtlas2

Figure 4 shows ForceAtlas2's 43rd timestep betweenness centrality. Again, the five nodes are larger than any others, and all other nodes not connected to one of the five are far away and out of view. This shows the graph's dense sub-graph and community.

V. Data preprocessing for the utilized Graph Convolutional Network approach

In the section, the data preparation procedure for our GCN-based model. Our data is divided into classes, features, and edge lists, as described above. The original dataset class file contains node IDs and class labels, while the edge list file contains two columns representing Bitcoin flow from one transaction node to the next. The feature file lists each node's timestep and features. The features include 93 handcrafted, original features and 72 embedding features from the original paper [3]. We use all features for our baseline anomaly classification algorithms, but only the original 93 for the GCN-based model. Due to the use of convolution-enhanced features, our baseline algorithms achieved better evaluation metrics than our GCN-based model.

The class file has three string types: '1' for illicit, '2' for licit, and 'unknown' for unclassified nodes, which were mapped to integers. Specifically, 0 was used for licit, 1 for illicit, and 2 for unknown. Then, we merged processed classes, timesteps, and features.

We extracted timesteps from the unified dataframe and selected edge list values and features for each timestep. We reset node IDs to zero for each time step to avoid integer overflow during training. Additionally, we applied a boolean classification mask to select nodes with known classes, excluding unknown class nodes from backpropagation training.

Each timestep was returned as a PyTorch Geometric Data object in a dictionary. Similar to the original paper, we selected 34 timesteps for training and 15 for testing.

VI. Our Approach

Our main approach is composed of two parts. In the first part, we utilize some baseline machine learning approaches, namely, a logistic regression model, a random forest classifier, and a multilayer perceptron classifier, in order to predict the binary dependent variable i.e., kind of transaction (licit or illicit), and evaluated the performance of the aforementioned algorithms using evaluation metrics, namely, precision, recall, f1-score, micro and macro f1-score. In the second part we employed a graph convolutional network architecture composed of three convolutional layers and evaluated them with the aforementioned metrics in to order to construct an overall comparison of the two parts for our overall approach.

A. Baseline algorithms

From the various baseline machine learning approaches, we employed the logistic regression model, the random forest classifier, and the multi-layer perceptron classifier to evaluate the classification of user-to-user transactions in the two aforementioned categories, namely, fraud, or illicit, and non-fraud, or licit. The logic behind the selection of these specific algorithms was to start with a relatively simple classifier i.e., the logistic regression model, and then employ more powerful ones, like the random forest classifier and the multi-layer perceptron, in order to evaluate how well they fit to our dataset, as well as the goodness of the achieved results. In all three evaluated classifiers, we dropped the unknown class observations and converted our initial multiclass classification to a binary classification problem to focus on capturing the malicious behaviors.

B. Graph Convolutional Networks

GCNs, introduced in [11], and originally designed for node classification in undirected, attributed, static graphs, can easily be employed for classifying anomalous nodes in a binary classification problem. GCNs perform convolutions on the attribute vectors of neighboring nodes, leaving graph structure intact, and capture high-order node relationships and interactions. Due to their nature, GCNs approach the node classification problem using a representation learning system,

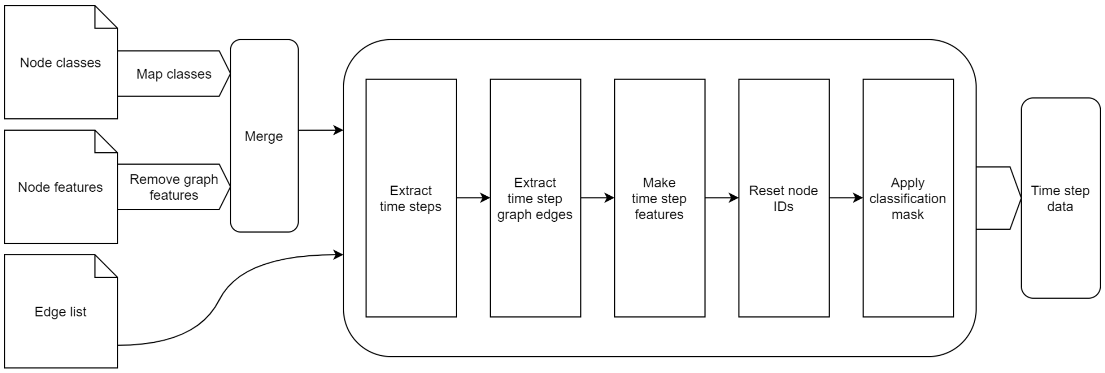

Fig. 5. The data preprocessing procedure for our GCN approach

capturing increasingly complex nodal attribute interactions at each layer.

In GCNs, the output of each layer is calculated as such [1]:

$$H^{(l+1)} = f(H^{(l)}, A|W^{(l)}) = \sigma(D'^{-1/2}A'D'^{-1/2}H^{(l)}W^{(l)}) \tag{1}$$

where:

- l denotes the layer,
- σ is a nonlinear function, such as ReLU,
- A' is the adjacency matrix plus the identity matrix, also counting self-loops,
- D' is the diagonal degree matrix of A', where $D'_{i,i}$ is the degree of the i-th node, and
- $H^{(0)}$ is the attribute matrix X.

The $D'^{-1/2}A'D'^{-1/2}$ operation is the symmetric normalization of matrix A'.

We can easily deduce that if $D'^{-1/2}A'D'^{-1/2}$ operation produces a matrix of $n*n$ dimensions, H is an $n*\gamma$ matrix and W is a $\gamma*\gamma'$ matrix, γ and γ' determine the attribute representation vector dimension of layers l and $l+1$.

Since the $D'^{-1/2}A'D'^{-1/2}$ operation is the same for all layers, it can be calculated as a preprocessing step.

In the above definition we assume equal importance of self-connections vs. edges to neighbouring nodes. It is worth noting we can assign self-loops a different weight, increasing or decreasing their importance. For some datasets, it can be beneficial to introduce a learned parameter, which plays a similar role as the trade-off parameter between supervised and unsupervised loss in the typical semi-supervised setting. In that case, the adjacency matrix is given as:

$$A' = A + \lambda i \tag{2}$$

where λ is the learned trade-off parameter [11].

Literature shows that a two- or three-layer network suffices and that further increase in depth has a negative impact in classification accuracy [11]. Stochasticity in the training process can be introduced via dropout for better results.

Residual, or skip, connections can also exist between layers, making deeper networks more accurate, with the output of each layer described as such:

$$\begin{aligned} H^{(l+1)} = f(D'^{-1/2}A'D'^{-1/2}H^{(l)}W^{(l)} + \\ H^{(l-1)}W^{(l-1,l)} + \cdots + H^{(0)}W^{(0,l)}) \end{aligned} \tag{3}$$

where $W^{(i,j)}$ describes a weight matrix for a skip connection from layer i to layer $j+1$, with $i<j$.

At the final layer, we use softmax instead of ReLU and evaluate the cross-entropy loss over all labeled examples.

Two-layer GCN example

To give an example, we describe the following network:

Assume a two-layer GCN with dropout of 0.5 between layers, n nodes, input attribute vector of dimension 32, layer embeddings of dimensions 24 and 16, and a residual connection from the input layer to the second layer. This can be described as such:

$$\begin{gathered} Z = H^{(2)} = \sigma(D'^{-1/2}A'D'^{-1/2}H^{(l)} + H^{(0)}W^{(0,l)}) = \\ \sigma(D'^{-1/2}A'D'^{-1/2}Drop_{0.5}(ReLU(D'^{-1/2}A'D'^{-1/2} \\ XW^{(0)}))W^{(l)} + XW^{(0,l)}) \end{gathered} \tag{4}$$

where:

- σ is a softmax function,
- $H^{(2)}$ or Z is the output of the network and the class probability vector,
- $H^{(l)}$ is the output of the first layer with size $n*24$,
- H^0, or X, is the input of the first layer with size $n*32$,
- $W^{(0)}$ is the weight matrix of the first layer and the attribute matrix with size $32*23$
- $W^{(l)}$ is the weight matrix of the second layer with size $24*16$ and
- $W^{(0,l)}$ is the weight matrix of the residual connection from the input layer to the second layer with size $32*16$.

Given the above example, and a sparse representation of the adjacency matrix A, the computational complexity of the network is $O(|E|*n*32*24*16+|E|*n*32*16)$ where $|E|$ is the number of edges.

In this context, it is also worth noting that a two-layer GCN with a skip-connection is equivalent to logistic regression when $W^{(0)}$ and $W^{(l)}$ are zero [3].

This way, one can further imagine how a three-, four-, and so on, layer network can be implemented.

Our proposed model architecture

When compared to the approach used in [11], we decided to implement a deeper model, resolving classification issues arising from model depth by applying residual connections between convolutional layers.

Our model architecture consists of three convolutional layers, followed by a linear layer that ends with a sigmoid output, instead of softmax, that addresses our binary classification problem. Due to this, we use the binary cross-entropy variation as our model loss during training. Our approach was inspired by Euclidian-space CNNs, where after all convolutions are performed, a linear layer is applied as the final classifier. Hence, one can think of as the initial convolutional layers having representational abilities, where at each layer node embeddings represent nodes in a different manner, and the final linear layer as a logistic regression classifier, using the final linear layer as a logistic regression classifier, using the final node embedding representation vectors as input.

Node embeddings dimensions are 64, 32 and 16 for each convolutional layer, respectively, and between convolutional layers we apply dropout with a rate of 0.2. Our approach was implemented in PyTorch, using the PyTorch Geometric extension, which is specifically built for Graph Neural Networks (GNNs).

We implemented three different network variations:

- A simple variation that implements no residual connections.
- A residual variation where each convolutional layer is connected with the layer before its previous one.
- A dense variation where each convolutional layer is connected to every previous layer.

So, given our previous two-layer GCN example definition, our dense network variation can be similarly described as such:

$$\begin{gathered} f(X,A) = sigmoid(H^{(3)}W^{(3)}) \\ H^{(3)} = Drop_{0.2}(ReLU(D'^{-1/2}A'D'^{-1/2}H^{(2)}W^{(2)} \\ +H^{(1)}W^{(1,2)} + H^{(0)}W^{(0,2)})) \\ H^{(2)} = Drop_{0.2}(ReLU(D'^{-1/2}A'D'^{-1/2}H^{(1)}W^{(1)} \\ +H^{(0)}W^{(0,1)})) \\ H^{(1)} = Drop_{0.2}(ReLU(D'^{-1/2}A'D'^{-1/2}H^{(0)}W^{(0)})) \\ H^{(0)} = X \end{gathered} \tag{5}$$

VII. RESULTS

This section presents our overall framework's key findings and variations. Table 1 provides the results. The logistic regression model has tuned with the default scikit-learn 0.24.1 parameters and achieved significant results due to the fact we extracted the majority of the dataset, i.e., the unknown class, keeping our dataset relatively small and easily manageable from simple machine learning approaches as the logistic regression. Using Adam solver and ReLU activation function improved the Multi-Layer Perceptron model. The Random Forest classifier, tuned with 50 trees in the created forest and a maximum depth of 100, was the best of the three baseline machine learning approaches. In all three models, the micro f1-score improves slightly from the baseline. Our GCN-based classifiers were trained for 275 epochs using the Adam optimizer, 0.001 learning rate, and 0.0005 decay rate. Our proposed models were largely outperformed by all baseline models, which we attribute to the use of convolutional embedding features that were discarded during data preprocessing for our GCN-based models. Our more sophisticated architecture helped us achieve slightly better evaluation metrics with fewer epochs than the original paper.

Approach	**Precision**	**Recall**	**F1 /Micro-F1**
Logistic regression	0.853	0.770	0.809/0.966
Random forest	0.999	0.885	0.939/0.989
Multi-layer perceptron	0.890	0.889	0.889/0.979
GCN	0.456	0.376	0.412/0.93
Residual GCN	0.727	0.566	0.636/0.958
Dense GCN	0.813	0.633	0.712/0.966
Skip-GCN from [11]	0.812	0.623	0.705/0.966

TABLE I
COMPARISON OF THE EVALUATION METRICS OF OUR PROPOSED FRAMEWORK WITH THE SIMILAR STATE-OF-THE-ART WORKS

Dense GCN outperformed all other GCN-based model variations by a wide margin, followed by residual and simple GCN. Thus, our results confirm the importance of residual connections in GCN-based classifiers.

Metric	**Illicit**	**Licit**
Average Degree	1.366	1.826
Avg. Weighted Degree	1.366	1.826
Network Diameter	4	24
Graph Density	0.034	0.001
Modularity	0.901	0.92
Connected Components	13	259
Avg. Clustering Coefficient	0	0.008
Avg. Path Length	1.698	6.965

TABLE II
COMPARISON OF THE GRAPH METRICS OF ILLICIT AND LICIT GRAPHS FOR THE 43RD TIMESTAMP.

Gephi can extract knowledge from graphs without code. We could then compare the illicit and licit graphs for timestep 43.

So, the above statistics provide us with some useful information about the:

- **Average Degree**. The average degree is simply the average number of edges per node in the graph. We can calculate it by dividing the total edges by the total nodes. This metric shows that the Licit graph has more edger per node and thus, this graph is stronger connected to the Illicit.
- **Average Weighted Degree** is the average of the sum of weights of the edges of nodes. It provides almost the same information as the Average Degree.
- **Network Diameter** is the maximum distance between any pair of nodes in the graph.
- **Graph Density** measures how close the network is to completion. A complete graph has all possible edges and densities equal to 1.

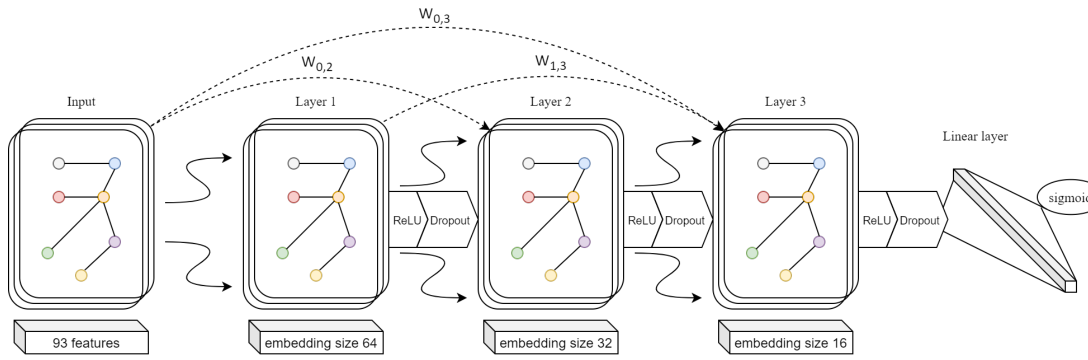

Fig. 6. Our four-layer, dense, GCN-based, proposed model architecture.

- **Modularity** is one measure of the structure of networks or graphs. It was designed to measure the strength of the division of a network into modules (also called groups, clusters, or communities). Networks with high modularity have dense connections between the nodes within modules but sparse connections between nodes in different modules.
- **Connected Components**. A connected component (or just component) of an undirected graph is a subgraph in which any two vertices are connected to each other by paths, and which is connected to no additional vertices in the supergraph.
- **Average Path Length** is defined as the average number of steps along the shortest paths for all possible pairs of network nodes. It is a measure of the efficiency of information or mass transport on a network.
- **Average Path Length** is defined as the average number of steps along the shortest paths for all possible pairs of network nodes. It is a measure of the efficiency of information or mass transport on a network.
- Local clustering of each node in the graph is the fraction of triangles that actually exist over all possible triangles in its neighborhood. The **Average Clustering Coefficient** of a graph is the mean of local clusters.

VIII. Conclusion

Detecting abnormalities is important for spotting rare events like outbreaks of rare disorders or side effects in medical diagnostics. Graphs capture long-range correlations between interdependent data objects. Graph-based anomaly detection is a new technology, with most papers published in the last decade. Combining Deep Learning and GCN models with graph-based models for anomaly detection is an undiscovered territory with a bright future. We designed a GCN-based anomaly detection system for bitcoin illegal transactions. Moreover, we compared machine-learning approaches for the same problem. Different architectures with different embedding dimensions can improve our GCN-based models. Despite our more sophisticated architecture, we outperformed the original paper's two-layer GCN model. Add residual connections to the final linear layer, or use dilated convolutions and deeper residual architectures, as shown in [12]. Recalculating the adjacency matrix and considering the cosine similarity between node attribute vectors can also improve performance [2]. A learned parameter can be used to give self-connections in adjacency matrix A' a different weight, as described in our GCN definition.

References

[1] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep anomaly detection on attributed networks," in *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 2019, pp. 594–602.

[2] J. Jiang, J. Chen, T. Gu, K.-K. R. Choo, C. Liu, M. Yu, W. Huang, and P. Mohapatra, "Anomaly detection with graph convolutional networks for insider threat and fraud detection," in *MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM)*. IEEE, 2019, pp. 109–114.

[3] M. Weber, G. Domeniconi, J. Chen, D. K. I. Weidele, C. Bellei, T. Robinson, and C. E. Leiserson, "Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics," *arXiv preprint arXiv:1908.02591*, 2019.

[4] T. Fawcett and F. Provost, "Adaptive fraud detection," *Data mining and knowledge discovery*, vol. 1, no. 3, pp. 291–316, 1997.

[5] S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. Subrahmanian, "Rev2: Fraudulent user prediction in rating platforms," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018, pp. 333–341.

[6] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos, "Fraudar: Bounding graph fraud in the face of camouflage," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 895–904.

[7] S. Ghosh and D. L. Reilly, "Credit card fraud detection with a neural-network," in *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*, vol. 3. IEEE, 1994, pp. 621–630.

[8] L. Zheng, Z. Li, J. Li, Z. Li, and J. Gao, "Addgraph: Anomaly detection in dynamic graph using attention-based temporal gcn." in *IJCAI*, 2019, pp. 4419–4425.

[9] M. Jacomy, T. Venturini, S. Heymann, and M. Bastian, "Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software," *PloS one*, vol. 9, no. 6, p. e98679, 2014.

[10] T. M. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Software: Practice and experience*, vol. 21, no. 11, pp. 1129–1164, 1991.

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

[12] G. Li, M. Müller, G. Qian, I. C. D. Perez, A. Abualshour, A. K. Thabet, and B. Ghanem, "Deepgcns: Making gcns go as deep as cnns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.