Exploration of Attention Layers with Evolving Graph Convolutional Networks for Dynamic Graphs for Detection of Fraudulent Activity in Bitcoin

Rachel Bosko
DS 440 Capstone
College of IST
University Park, PA
rtb5300@psu.edu

***Abstract*— Graph convolutional networks (GCNs) are state-of-the-art deep learning approaches for semi supervised graph-structured data. Recently, GCNs and similar models have shown superior performance in various application areas on real-world datasets, specifically the detection of fraud. In this paper, we study the problem of over-smoothing and analyze the stacking of layers for deep graph convolutional networks. We propose the implementation of 1, 2, 3, and 4 layers. Our experiments show that the problem of over-smoothing occurs after 3 layers; there is no significant difference between the use of 2 versus 3 layers.**

I. Introduction

Cryptocurrency is a form of digital currency that generally only exists electronically. Paying with cryptocurrency has drawbacks relating to protection compared to paying with a credit card or other traditional payment methods. Problems include: payments do not come with legal protection, transactions are typically not reversible, and some information about the transaction may be public. Often people relate cryptocurrency to transactions being anonymous, but that is not always the case. Some cryptocurrencies record some transaction details on a blockchain, or a public decentralized ledger. Information included in these details can be used to identify a person. Before exchanging cryptocurrency, both parties should know each other's reputation to avoid fraudulent activity.

Graph convolutional neural networks (GCN) is an extension of convolutional neural networks (CNN). Traditional GCNs operate by converting the graph structure into an adjacency matrix, and then use it as a collection or gather operation into select and average one-hop neighborhoods. GCN is considered state-of-the-art for graph structures in a static setting, but graphs dynamically evolve as we approach more practical scenarios. Existing methods resort to node embeddings and use a recurrent neural mechanism to regulate the embeddings and learn the temporal dynamics. This requires the knowledge of the node in the full time span and is less pertinent to the frequent change of the node set.

EvolveGCN, developed by MIT-IBM Watson lab, proposes to resolve this challenge to adapt the GCN model along the temporal dimension without resorting to node embeddings by using RNN to evolve the GCN parameters. Since Bitcoin transactions are constantly flowing this is a more applicable approach to accurately detect fraudulent transactions in real time. To extend the dynamic setting, I propose to explore the use of a different number of layers.

II. Related Work

A. Deep Network Representation Learning

The paper "Bitcoin Transaction Forecasting with Deep Network Representation Learning" presents a new approach to developing a Bitcoin transaction model, DLForecast, by leveraging deep neural networks for learning the transaction network of Bitcoin. Their DLForecast makes three original contributions. First, the exploration of transaction amount patter, transaction dynamics, and topological connectivity pattern of Bitcoin accounts. Second, they created a time-decaying transaction pattern graph and reachability graph, which is intended to capture different kinds of spatial-temporal transaction patterns of Bitcoin. Third, they develop a Bitcoin transaction forecasting system between user accounts using historical transactions by implementing node embedding on both graphs [1]. The use of Graph Convolutional Network (GCN) and transaction prediction is beyond the scope of this paper.

B. REV2

The paper "REV2: Fraudulent User Prediction in Rating Platforms" presents a system to identify fraudulent users. They propose three interdependent properties: fairness of a user, goodness of a product, and reliability of a rating. They developed an algorithm, REV2, to calculate properties for all users, products, and ratings by combining behavior and network properties. The authors claim that the algorithm converges and has a linear time complexity [2]. Currently this algorithm outperforms nine existing algorithms in fraudulent users on the market and is currently deployed at Flipkart, an e-commerce company.

C. Edge Weight Prediction

The paper "Edge Weight Prediction in Weighted Sign Networks" uses weighted signed networks (WSNs) to capture trust or distrust, like or dislike, and other social relationships. This paper attempts to predict the weights of edges in such networks proposing two novel measures of node behavior. First, the goodness of a node recognizes how much a node is trusted or liked by other nodes. Second, the fairness of a node recognizes how fair the node is in assessing other nodes' trust and likeability. The mutually recursive definition the authors created converges to a unique solution in linear time of the two concepts. This has not been done in past work [3]. Fairness and goodness metrics are used in various regression models and proves to predict edge weights on different types of networks like Bitcoin, Wikipedia, Twitter, and more with accurate results.

D. Graph Convolutional Networks

The use of Graph Convolutional Networks (GCNs) for financial forensics including anti-money laundering in Bitcoin is considered state-of-the-art. In the paper "Fraud Detection: A Systematic Literature Review of Graph-Based Anomaly Detection Approaches.", the authors use a variety of machine learning techniques to the given datasets. The best results to support their hypothesis on the nature of illicit transactions from this paper was obtained using DeepWalk. Deepwalk is a fairly new process that learns the representation of graphs by walking on vertices. This is definitely something to note to possibly use for our project to generate embeddings of the graph. The author concludes, "DeepWalk helped us to achieve our best results and further our hypothesis on the nature of illicit transactions. Although Random Forest performed better in terms of f1 score, graph features proved to be better identifiers of illicit transactions and SVM performed much better on the embeddings because the data is sparse making it harder for the Random Forest model to find distinctions in the features" [3]. Embeddings computed by Deepwalk that were trained on SVM gave an accuracy of 95% which is the highest accuracy they received compared to other algorithms. Better performance using SVM can be explained by its ability to transform the data into a higher dimension where the difference between classes is more clear. In this paper the use of graph convolutional networks is hypothesized to better generate graphs that represent illicit transactions, but is not performed.

III. Methodology

A. Data

The Elliptic Dataset maps Bitcoin transactions to real entities belonging to licit class (miners, wallet providers, exchanges, licit services, etc.) versus illicit class (malware, scams, ransomware, terrorist ogranizations, etc.). The Elliptic dataset nodes represent a transaction, and the edge is a "flow of bitcoins" between transactions. Each node has 166 features and is in fact labeled by "licit", "illicit", or "unknown". This dataset is an anonymized transaction graph collected from the Bitcoin blockchain.

The graph consists of 203,769 nodes and 234,355 edges. Two percent of the nodes are labeled illicit, class1. Twenty-one percent are labeled licit, class 2. The remaining graph transactions are not labeled.

B. Graph Convolutional Network (GCN)

A GCN consists of multiple layers of graph convolution, this is similar to a perceptron but has an additional aggregation step prompted by spectral convolution. Given a graph G = (V, E), a GCN takes as input

- an input feature matrix $N \times F^0$ feature matrix, X, where N is the number of nodes and F^0 is the number of input features for each node
- an $N \times N$ matrix representation of the graph structure such as the adjacency matrix A of G [4]

A hidden layer in the GCN can thus be written as $H^i = f(H^{i-1}, A))$ where $H^0 = X$ and f is a propagation [4]. Each layer H^i correlates to an $N \times F^i$ feature matrix where each row is a feature representation of a node. These features are aggregated at each layer to form the next layer's features using the propagation rule.. Thus, at each consecutive layer features become increasingly more abstract. [4]

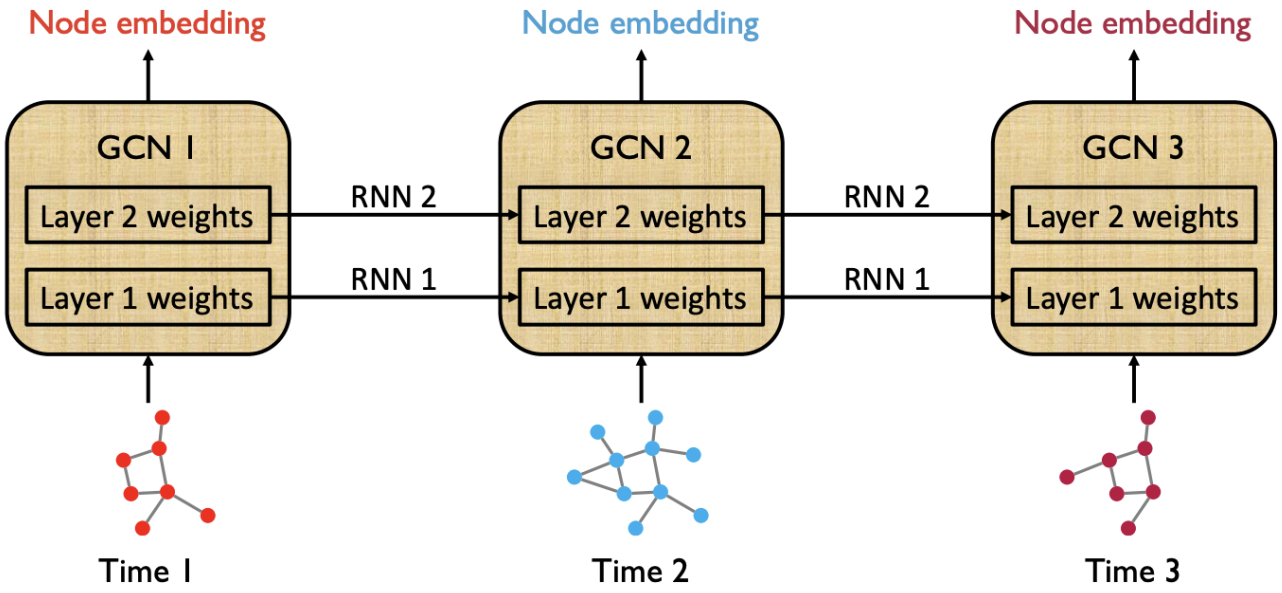

Figure 1: Schematic illustrations of EvolvGCN. The RNN refers to a recurrent architecture, in this case LSTM [5].

C. Weight Evolution

The main proposed approach of EvolveGCN updates the weight matrix at time based on current, as well as historical information. This requirement is substituted by using a recurrent architecture. For my research purposes I focused on their second option to treat the weight matrix as the output of the dynamical system. This uses a long short-term memory (LSTM) cell to model this input-output relationship. By using a cell context the LSTM is able to maintain the system's information. Please note in this version node embeddings are not used at all. For future work it is possible to replace LSTM by using another recurrent architecture. This version of the evolving graph convolution network is referenced as EGCN-O.

D. Evolving Graph Convolution Unit (EGCU)

Combining a graph convolutional network and a recurrent architecture creates the evolving graph convolution unit (EGCNU). For the purpose of my research we will focus on one of the two versions.

1: function Ht = f(Xt)
2: Current input Xt is the same as the past output Ht−1
3: Ft = sigmoid(WF Xt + UF Ht−1 + BF)
4: It = sigmoid(WIXt + UIHt−1 + BI)
5: Ot = sigmoid(WOXt + UOHt−1 + BO)
6: Cet = tanh(WCXt + UCHt−1 + BC)
7: Ct = Ft ◦ Ct−1 + It ◦ Cet
8: Ht = Ot ◦ tanh(Ct)
9: end function

[5]

Along with the above function *f*, specify the recurrent architecture. Note that LSTM is used, but can be replaced with another RNN architecture.

W (l) t = LSTM(W (l) t−1) := f(W (l) t−1).

E. Novelty

Implementation of EGCN-O requires only a straightforward extension of the standard LSTM from the vector version to the matrix version.

My novel approach is to explore the stacking of a different number of layers of this model and evaluate. By adding more layers, these features are aggregated to form the next layer's features using the propagation rule. The graph convolutional layer represents each node as an aggregate of its neighborhood. My code implements the use of 1, 2, 3, and 4 layers and explores the effect of a different number of layers. It is important to note that GCNs are very shallow models due to the vanishing gradient descent problem and should not go past 3 or 4 layers. For this case, I looked for over-smoothing by observing the validation metrics.

F. Experiments

In this section, we present a set of experiments to demonstrate the effectiveness of a different number of layers of EvolveGCN.

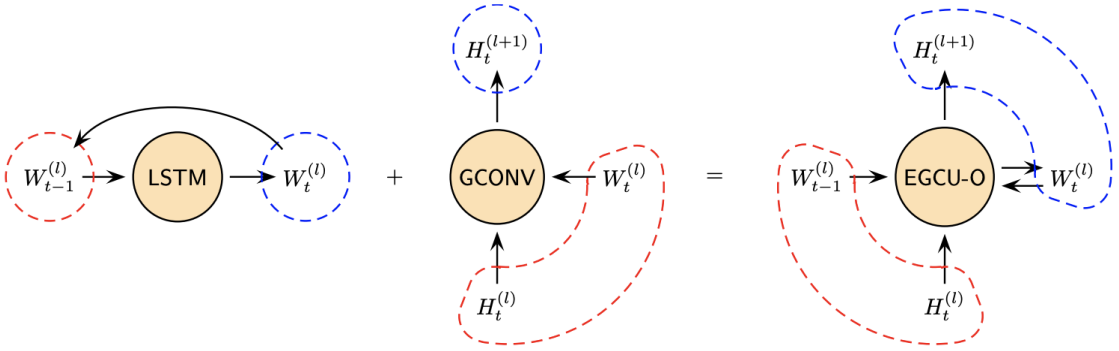

Figure 2: EvolveGCN-O, where the GCN parameters are inputs/outputs of a recurrent architecture [5].

All experiments were run on ACI-b Standard Core without GPU acceleration with ICDS-ACI. Experiments were normally run overnight to train the model since it takes many hours without GPU acceleration. Test results are reported at the best validation epoch and hyperparameters are tuned by using the validation set. We use a learning rate of 0.001 and early stopping with a patience of 100. For the elliptic dataset we split the nodes of each class into 65%, 10%, and 25% for training, validation, and testing.

G. Tasks

The EvolveGCN proposes three predicted tasks.

1) *Link Prediction*
2) *Edge Classification*
3) *Node Classification*

IV. Evaluation

A. Metrics

The elliptic dataset classes correspond to licit and illicit transactions respectively and are highly skewed. The illicit class (minority) is the main interest in terms of financial crime forensic. In this case, we care more about detecting illicit cases using recall while keeping the cost at which it is achieved under control, precision. The precision-recall tradeoff is a very challenging problem with imbalanced datasets, in the case of the detection of fraud I have prioritized recall.

B. Results

2-Layers

Precision	Recall	f1
0.1133	0.9698	0.2029

3-Layer

Precision	Recall	f1
0.1358	0.9486	0.2375

4-Layers

Precision	Recall	f1
0.1504	0.8173	0.2541

The results of my experiments display that as we stack more layers and the model becomes deeper, recall decreases while precision very slightly increases. There is no statistically significant difference between using 2 or 3 layers. But, it appears that when we use 4 layers in our model recall drops and it seems like the model is starting to over-smooth and become too deep.

V. Discussion

GPU availability is highly recommended to train the models. I have ran all my experiments on ACI-b Standard Core without GPU acceleration with ICDS-ACI, but this takes many hours to train my model and it would be beneficial to get remote GPU server access. I received access to a GPU-enabled server, but was unable to be given sudo access since IST Help Desk never responded. My code runs on Python 3.6, which is a deprecated version of Python and I was required to switch versions but was unable to since I did not have admin access. Hopefully students in the future will be able to switch to different versions, even deprecated, to properly work on the GPU servers.

In the future, it would be beneficial to explore replacing the GCN model with a different newer state-of-the-art model. I ran out of time, but I did explore the opportunity to replace the GCN model in this code with a Graph Attention Network (GAT) model. GAT is a neural network architecture that operates on graph-structured data, to address the shortcomings of prior methods like GCN based on graph convolutions and their approximations the

model leverages masked self-attentional layers. The model stacks layers in which nodes are able to attend over their neighborhoods' features, this enables specifying different weights to different nodes in a neighborhood without needing any sort of expensive operation. Similar to my current model, I would use RNN to evolve the GAT parameters to further solve the problem of the dynamic setting.

In addition, the precision score of my model is very low. The aim of my model was to have a high recall score to detect fraudulent cases, giving a lower number of false negatives. Unfortunately, I cannot have both precision and recall very high due to the precision-recall tradeoff. It would be beneficial for someone in the future to explore options to combat this and improve the precision score of my model while keeping the recall score relatively high.

VI. Conclusion

In conclusion, there is no significant difference in results between using 2 vs 3 layers for an evolving graph convolutional network. At 4 layers, it appears that the model begins to over-smooth and become too deep due to the vanishing gradient problem. While this is not the result I hoped for, I learned there in this case there is no need to make the model more complex with more layers. Shallow architectures limit the model's ability to extract information from high-order neighbors. As the number of layers increases, the representations of nodes in GCN begin to converge to a certain value, known as over-smoothing. A 2-layer GCN model outperforms deeper GCN models.

VII. Contribution

The original code is from EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs, published in AAAI 2020 by Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Time Kaler, Tao B. Shardl, Charles E. Leiserson from MIT-IBM Watson AI Lab, IBM Research, and MIT CSAIL.

My final project was completed solely by myself, Rachel Bosko. My novel contribution to the original code base is adding a different number of layers of graph convolution from the original proposed 2-layers. I ran experiments with 1 to 4 layers, and can be referenced in my github repository with corresponding names.

In addition, I want to give a special thank you to Suhang Wang for meeting with me for a consultation on neural networks, specifically Graph Convolutional Networks (GCN). His key research areas are in the area of data mining, social media mining, machine learning, and deep learning. After my meeting with him we were able to come up with possible novel solutions I could add to my project.

Appendix

[1] Wei, W., Zhang, Q., & Liu, L. (2022, March 8). *Bitcoin transaction forecasting with Deep Network Representation Learning*. arXiv.org. Retrieved from https://arxiv.org/abs/2007.07993

[2] Kumar, S. (n.d.). *REV2: Fraudulent user prediction in rating platforms*. Retrieved from https://faculty.cc.gatech.edu/~srijan/pubs/rev2-wsdm18.pdf

[3] Kumar, S. (n.d.). *Edge weight prediction in weighted signed networks*. IEEE Xplore. Retrieved https://ieeexplore.ieee.org/document/7837846

[4] Jepsen, T. (2019, May 7). *How to do deep learning on graphs with graph convolutional networks*. Medium. Retrieved from https://towardsdatascience.com/how-to-do-deep-learning-on-graphs-with-graph-convolutional-networks-7d2250723780

[5] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumara, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. AAAI 2020.

[6] GitHub repo: https://github.com/rbosko/DS440.git