Fraud detection with Graph Attention Networks

Rohith Yamsani
Department of Knight Foundation School of Computing and Information Sciences
Florida International University
ryams001@fiu.edu

Sachin Sravan Kumar Komati
Department of Knight Foundation School of Computing and Information Sciences
Florida International University
skoma004@fiu.edu

***Abstract*— Fraud detection has become a major issue with the financial services industry's explosive growth in order to ensure a safe environment for both users and suppliers. The majority of conventional solutions for fraud detection rely on rule-based techniques or manually divert some features for prediction. However, consumers of financial services engage in rich interactions and consistently display a variety of information. These data provide a sizable multiview network that is underutilized by current techniques. Additionally, only a tiny fraction of the network's users is labeled, which presents a significant obstacle for relying solely on labeled data to successfully detect fraud.

We propose multi-view labeled and unlabeled data for fraud detection as a solution, expanding the labeled data through their social links to obtain the unlabeled data. Additionally, we suggest a hierarchical attention technique to enhance the correlation between various neighbors and various viewpoints. The attention mechanism can simultaneously make the model interpretable, reveal the key components of fraud, and explain why users are anticipated to commit fraud. Our method can do two tasks more accurately than state-of-the-art methods thanks to the usage of social relations and user traits. Additionally, the interpretable outcomes offer intriguing intuitions about the activities.

Index Terms—Graph Attention Network, Fraud Prediction, Graph Neural Network

I. Introduction

Financial services, particularly those offered online, are quite convenient for many people today while also producing significant economic benefits for society. But we are also seeing an increase in financial fraud. For instance, a record 15.4 million customers in the US alone reported experiencing fraud, a 16 percent increase from 2015. Last year, fraudsters stole roughly 6 billion from banks1. There are many different types of fraud, including insurance fraud, default, and case-out fraud in credit-card services. All of these frauds substantially jeopardize the security of both users and service providers. Therefore, it is crucial to look into the issue of fraud detection. Predicting if an entity, which might be a user, a device, or more, will be involved in fraud is the aim of fraud detection.

The issue can typically be expressed as a categorization issue. There are two sorts of traditional fraud detection techniques. The first category includes rule-based approaches. They believe that by examining obvious and surface-level indications, financial fraud can be found. These signals can be used to develop various fraud prediction rules. Although rule-based approaches have been useful to the sector for a while, they are not without flaws. Rule designs rely primarily on prior information held by humans. As a result, it is challenging to manage the shifting and intricate patterns using these methods. Additionally, rule-based approaches are open to assault. Machine learning techniques are suggested to automatically mine the fraud tendencies from data in order to overcome their limitations. The majority of machine learning techniques extract statistical information about users from a variety of sources, including user profiles, user activities, and transaction summary. These techniques use conventional classifiers like logistic regression and neural networks to perform classification and base their predictions mostly on the statistical characteristics of a particular user. These approaches, however, hardly ever take user interactions into account. In reality, the financial scenarios have complex relationships. Users may be in social ties with one another, such as that of friends, classmates, and relatives. Users may conduct business with both merchants and other users. In some apps, logging in is required in order to complete financial activities. The fraud detection issue may benefit from all of these connections. Then, some of the solutions that come after start to use graph embedding to include user involvement. However, for fraud detection, only a small percentage of the data are labeled, and the majority of the data are typically unlabeled and have not yet been completely utilized by existing graph-based algorithms. In financial scenarios, interpretable models and outputs are frequently sought, yet most graph embedding techniques are black-box models.

We provide a graph attentive neural model for multiview data called the fraud detection with graph attention network to address the aforementioned three issues.

This project's main goal is to improve user representations by fully utilizing relational and attribute data from both labeled and unlabeled data. We specifically leverage the social connections between identified and unlabeled users to create a bridge. Additionally, we establish the attribute network for each user using his attributes. Together, the social connections and the qualities create a substantial multiview network. Then, in order to simultaneously represent the multiview information within the network for fraud detection, we propose a semi-supervised graph neural network. The model provides a number of benefits: (1) For fraud detection, it can completely utilize both supervised and unstructured structural information. (2) Our model may include multiview data to produce a thorough fraud detection result. (3) Creating an attribute network rather than using attributes as dense features can improve the ability of attribute information to be represented. A hierarchical attention mechanism is also something we develop. The first-layer node-level attention is made to efficiently correlate various users' properties or distinct neighbors. Different views of the same data can be correlated via the second layer of view-level attention. Additionally, the outcomes of the attention-based model may be interpretable and offer additional task-related insights.

II. Related Work

A. Financial Fraud Detection

Our work is related to the issue of fraud detection empirically. Financial fraud is a problem with far-reaching effects on both the financial sector and everyday life. As a result, several academic studies on various types of fraud, including financial statement fraud credit card fraud [10], [11], insurance fraud [12], [13], and so on, have been conducted. For fraud detection, earlier efforts mostly used rule-based techniques. They presume that there are some clear trends in the fraud activities. These studies define a few combinatorial rules to identify these fraud actions as a result . Rule-based methods are widely used for fraud detection since they are straightforward and easy to understand. However, rule-based approaches rely heavily on the knowledge of human experts. Complex and mutating patterns are challenging to locate. Additionally, if attackers are aware of the regulations, they are likewise more vulnerable to attack.

B. Learning over graphs

Our work is connected to graph-based approaches technically. Network embedding is a useful technique for simulating a graph's structure. For each node, it seeks to learn a low-dimensional vector representation. Early research is mostly concerned with the pure network without node properties.

III. GAT ARCHITECTURE

The building block layer used to create arbitrary graph attention networks will be introduced in this section, along with its theoretical and practical advantages, disadvantages, and potential applications in comparison to earlier work in the field of neural graph processing.

In this section, we will describe the single graph attentional layer that was used in all of the GAT architectures we tested. The specific attentional arrangement we use closely reflects the research of Bahdanau et al. (2015), although the framework is independent of the specific attention mechanism chosen.

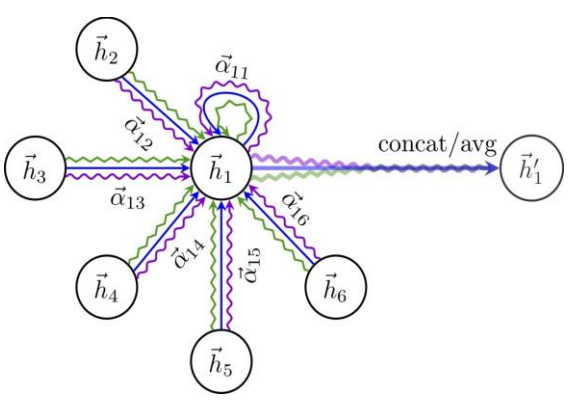
The input to our layer is a set of node features, $h = \vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N, \vec{h}_i$ *belongs to* R^F, where N is the number of nodes, and F is the number of features in each node. The layer produces a new set of node features (of potentially different cardinality F'), $\mathbf{h}' = \{\vec{h}'_1, \vec{h}'_2, \ldots, \vec{h}'_N\}, \vec{h}'_i \in R^{F'}$, as its output.

First, we outline the broad outlines of our issue. To solve this issue, we gather data from various perspectives, each of which represents a different aspect of the user's data. Like social relations and transaction relations, some views of the graph are formed by default. However, in addition to formulating some user attributes as attribute graphs rather than dense features, we do so as well. This improves the ability of our graph model to identify relationships between the traits, according to our findings. Notably, we are proposing a broad model here. However, we have a number of circumstances that several views, each of which has distinct information. In the experiment, we shall detail the dataset in detail. Our goal with the multiview network is to train a classifier to identify user fraud.

We summarize the notations of this paper in Table I. We have n_L labeled users denoted as U_L, each of which is labeledas fraud or not. From these users, we go through the social relations of the labeled users and thus get n_{uL} unlabeled users, denoted as U_{uL}. We have n_{uL} n_L. For each user, we have$m$ views of data. In each view, we have a view-specific graph denoted as G^v = U S^v, E^v , v 1, ...{m ∪Here S^v}denotes{ some view-specific nodes. For example, if we use the user-app graph, S^v should be the app set. If we use the relation graph, S^v may be empty. And if we use the attribute graph, S^v may be the attribute sets.

TABLE I
NOTATIONS AND EXPLANATIONS.

Notation	Explanation
$U = U_L \cup U_{UI}$	User Set
m	Number of views
n_v	Number of nodes in the v-th view-specific graph
y_u	the label of the user u
N_u^v	the neighbors of user u in the v-th view-specific graph
L	the number of layers for the view-specific MLP



We design a hierarchical attention structure in graph neural network: from node-level attention to view-level attention to integrate multiple neighborhoods and different views. The framework of the whole model can be shown in Figure 2. Firstly, we propose a node-level attention to learn the weight of neighbors for users in each view and accordingly aggregate the neighbors to obtain the low-level view-specific user embeding. Secondly, multiview data characterizes different facets of user information and thus it is essential to integrate multiview data. However, multiview raw data usually have different statistical properites which makes it difficult to assmeble multiviewdata in the low-level space. Since multiview data describethe same thing, they should show large semantic similarity. Meanwhile, the representations in the high-level space are more close to the semantic. Considering this, we use seperate models to project low-level view-specific user embedding into the high-level space and then integrate view-specific embedding to obtain the joint embedding. Specifically, we propose the view-level attention here to tell the difference of different views and get the optimal combination of view-specific user embedding for the task. Finally, with the combined embedding, we design the supervised classification loss and unsupervised graph-based loss to fully utilize the labeled and unlabeled data.

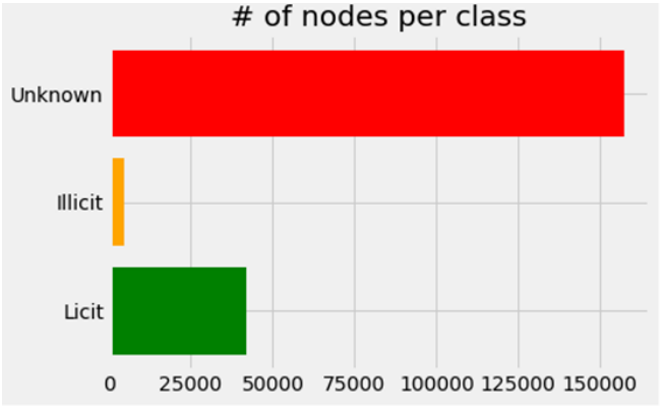
IV. Graph Dataset

We are using the Bitcoin dataset, also known as the Elliptic Data Set [1] [2], which associates real things with either legal or illegal Bitcoin transactions. This dataset is freely available and may be obtained from the Kaggle data science platform [3].

This consists of three csv files: the first one labels nodes (licit/illicit/unknown), the second describes edges between nodes using their "Ids," and the third contains node ids with 166 attributes.

A transaction is represented as a node in the graph, and a movement of bitcoins between transactions is represented by an edge. Each node has been classified as having either 166 traits and being the work of a "licit," "illicit," or "unknown" entity.

The graph has 203,769 nodes and 234,355 edges. As shown in Figure 2, 2% (4,545) of the nodes are classified as class 1 (illicit), while 21% (42,019) are classified as class 2 (licit). The remaining transactions are not differentiated between legal and illegal trades. 49 different time steps exist.



The first 94 features are local information about the transaction, and the final 72 features are aggregated features that are obtained using transaction information that has been moved one hop backwards or forward from the center node. These features provide the maximum, minimum, standard deviation, and correlation coefficients of the neighbor transactions for the same information data (number of inputs/outputs, transaction fee, etc.).

In order to determine if a node is legitimate or illegal, we will use this dataset to perform a node classification task.

The programming examples must be understood in order to be understood, Pytorch must be familiar.

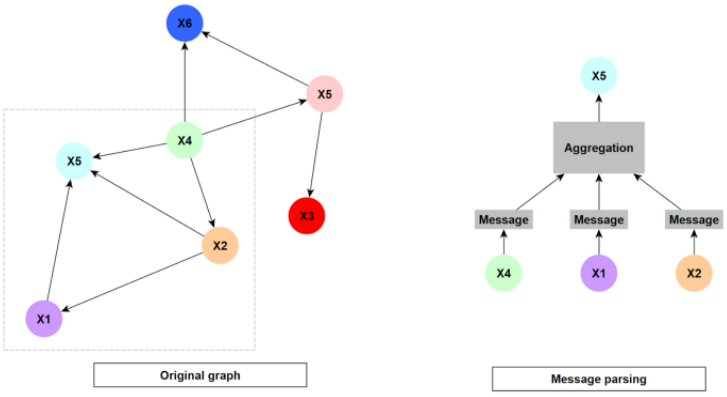
We'll assume you have a basic understanding of backpropagation, neural networks, and machine learning.

V. Graph Neural Networks

The popularity of Graph Neural Networks (GNNs) [8,9,10] is rising. GNNs are neural networks that can easily perform node-level, edge-level, and graph-level prediction tasks since they can be applied directly to graphs.

The domains of biology, chemistry, social science, physics, and many others have experienced notable advancements and triumphs in GNNs in recent years. On many benchmarks, it has resulted in state-of-the-art performance. The following graphic illustrates how GNNs perform message passing to aggregate data from neighbors and take into account both instance-level and graph-level properties when making choices. Great performance on this kind of task results from this.

Below image illustrates the mechanism of GNN



A graphical relationship is transformed into a system using a GNN in which data messages are transmitted from nearby nodes through edges and combined into the target node. The way that each node aggregates and mixes the representations of its neighbors with its own varies amongst the many GNN variations.

We first introduce the attention mechanism through the original GAT model for a task of fraud detection on Bitcoin transactions, and then we will demonstrate a second version dubbed GATv2.

Graph Attention Networks (GATs) are one of the most well-liked GNN architectures, and Velickovic et al. introduced them. GATs outperform other models on a variety of benchmarks and tasks (2018). Both of these versions make use of the "Attention" technique [5], which has had considerable success across a number of ML domains, including NLP with Transformers.

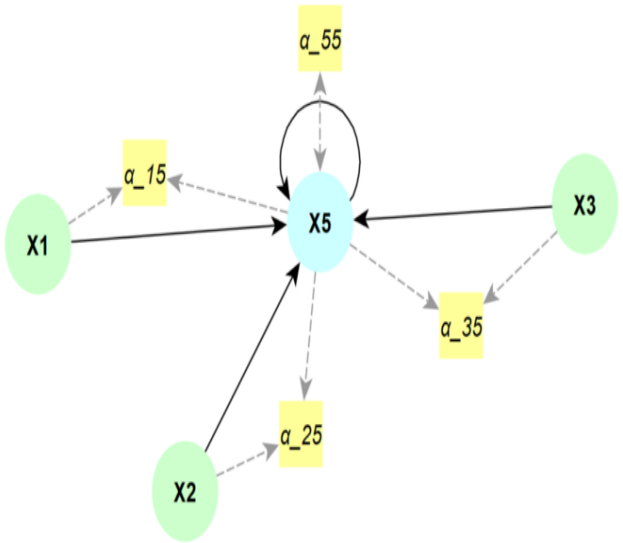
On top of Pytorch, the most well-liked graph deep learning framework, Pytorch Geometric PyG, will be used in this example. With a large number of graph models already constructed for a variety of applications involving structured data, PyG is suitable for fast implementing GNN models. Additionally, a training/evaluation pipeline can be created using GraphGym, which is included with PyG.

GAT v1

Many common GNN architectures, including GraphSAGE, give all neighbors' messages the same weight (e.g mean or max-pooling as AGGREGATE). While utilizing its own representation as the query, each node in a GAT model updates its representation by paying attention to its neighbors. As a result, each node calculates a weighted average of its neighbors and then chooses those that are most pertinent to it.

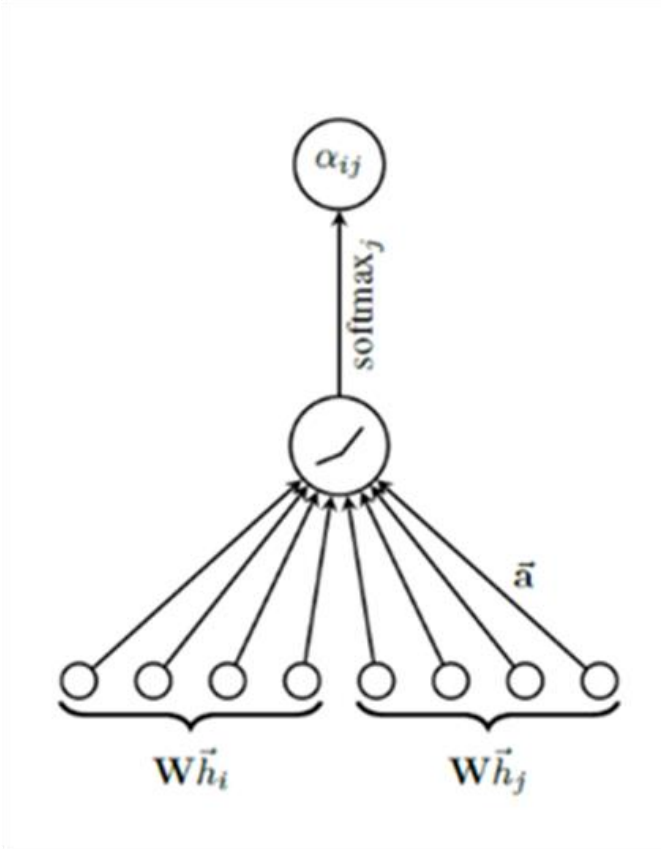
The next Figure, which depicts a single attention mechanism, illustrates how this model uses an attention mechanism (ij) to assess the significance of each message being passed by various nodes in the neighborhood.

Below figure illustrates the single attention mechanism of GAT.



The relevance of the features of the neighbor j to the node I where a shared attentional mechanism "a" and a shared linear transformation parametrized by the weight matrix "W" are learned, are computed using a scoring function e to determine the attention score between two neighbors.

Below figure is the illustration of attention mechanism with



weight vector.

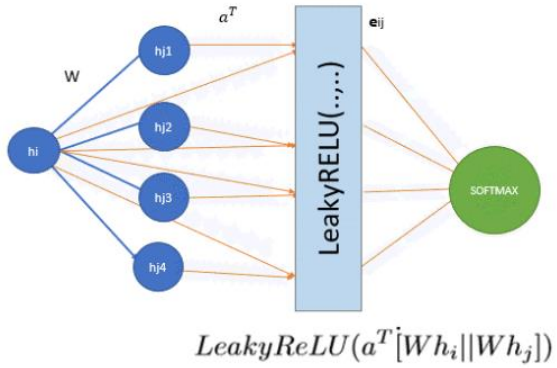
A single-layer feedforward neural network with a weight vector and a non-linear activation function will make up our attention mechanism "a." (LeakyRelu). We only calculate the attention coefficients eij for nodes j in the graph's immediate vicinity of node I

$$e(h_i, h_j) = LeakyReLU(a^T[Wh_i||Wh_j]) \quad (1)$$

Softmax is used to standardize these attention scores across all of Ni's neighbors. The attention function is established as follows by converting the attention coefficients into a probability distribution:

$$\alpha_{ij} = softmax(e(h_i, h_j) = \frac{exp(e(h_i, h_j)}{\sum_{j' \epsilon N_i}(exp(e(h_i, h'_j)} \quad (2)$$

The whole computations of this model can be visualized below:



$$LeakyReLU(a^T[Wh_i||Wh_j])$$

The neighbors' embeddings are then combined, scaled by attention scores based on the graph's structural characteristics (node degree). Using the normalized attention coefficients, the GAT model calculates a weighted average of the modified features of the neighboring nodes as the new representations of I

$$h_i^{(l)} = \sigma(\sum_{j \epsilon N(i)} \alpha_{ij} W^{(l)} h_j^{(l-1)}) \quad (3)$$

In contrast to a normal Graph Convolution Network / GraphSAGE where all messages are weighted identically, the attention mechanism has trainable parameters and is dynamic.

Multi-headed attention is another technique employed since it enhances performance, particularly when stabilizing learning. It broadens the network's expressivity by capturing various degrees of relevance for nodes in the same neighborhood.

On the final (prediction) layer of the network, K independent attention mechanisms transform the previous Equation (3) because concatenation is no longer sensible with || a concatenation operation, as shown in the following Equation (4) when using GAT with multi-headed attention.

$$h_i^{(l)} = ||_{k=1}^{K} \sigma(\sum_{j \epsilon N(i)} \alpha_{ij}^k W^{(k)} h_j^{(l-1)}) \quad (4)$$

instead, we use **averaging**, we get the following formula:

$$h_i^{(l)} = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in N} \alpha_{ij}^k W^{(k)} h_j^{(l-1)}) \quad (5)$$

As a result, the GAT model computes a static attention in its first implementation, where the ordering of attention coefficients is global for all nodes in the graph, shared by all nodes in the graph, and independent of the query node. This strategy is constrained and affects the GAT's expressiveness because the attention function is monotone with respect to the neighbor (key) scores.

The future work may focus on differentiating different social relations to further improve the model. And we can extend the model to more fraud detection applications.

CONCLUSION

In order to increase the model's capacity, Graph Attention Network assigns varied priorities to nodes in the same neighborhood while still working on all of the surrounding nodes.

In this we demonstrate a thorough implementation of GAT as well as the upgraded GATv2, a more expressive implementation that makes use of dynamic attention by altering the sequence of operations and is more resistant to noisy edges.

The prebuilt GAT models outperform previous benchmarks utilizing GCN, outperforming them both by 0.92 F1 macro and 0.97 accuracy on the fraud dataset. The authors argue that the GATv2 model should be used as a baseline instead of the original GAT model because it outperforms GAT on numerous benchmarks.

REFERENCES

1. Elliptic website, www.elliptic.co.
2. Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics, arXiv:1908.02591, 2019.
3. M. Weber G. Domeniconi J. Chen D. K. I. Weidele C. Bellei, T. Robinson, C. E. Leiserson, https://www.kaggle.com/ellipticco/elliptic-data-set, 2019
4. Graph Attention Networks, Petar Veličković and Guillem Cucurull and Arantxa Casanova and Adriana Romero and Pietro Liò and Yoshua Bengio, arXiv:1710.10903, 2018
5. Attention Is All You Need, Ashish Vaswani and Noam Shazeer and Niki Parmar and Jakob Uszkoreit and Llion Jones and Aidan N. Gomez and Lukasz Kaiser and Illia Polosukhin, arXiv:1706.03762, 2017.
6. How Attentive are Graph Attention Networks?, Shaked Brody, Uri Alon, Eran Yahav, arXiv:2105.14491, 2021
7. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges, arXiv:2104.13478, M. M. Bronstein, J. Bruna, T. Cohen, P. Velickovic, 2021.
8. https://tkipf.github.io/graph-convolutional-networks.