LEVERAGING MULTI-TASK LEARNING GRAPH NEURAL NETWORKS FOR IMPROVING FRAUD DETECTION

by

Sirshendu Ganguly

Under the supervision of

Prof. Fabricio Murai, Advisor
Prof. Yanhua Li, Reader

DEPARTMENT OF DATA SCIENCE

WORCESTER POLYTECHNIC INSTITUTE

SPRING 2023

Contents

List of Figures

List of Tables

ABSTRACT

This thesis explores the challenges of detecting fraudulent activities such as money laundering detection in the financial ecosystems and forged review detection in e-commerce websites. One of the major differences between fraud detection and other classification problems is the class imbalance ratio. Class imbalance is a phenomenon that occurs when the number of examples in each class of a dataset is not evenly distributed, for example, the ratio between the number of illicit transactions and that of licit transactions in a fraud detection problem is very small. In this thesis, we explored three graph datasets commonly used for benchmarking fraud detection techniques, the Elliptic dataset, the fraud Amazon dataset, and the fraud Yelp dataset. Our goal is to increase the raw feature set by node embeddings generated by complementary tasks such as link prediction, and link classification before the final classification task. The current limitations of existing tools in accurately estimating fraud, along with the difficulties associated with detecting fraudulent activities in general, are discussed. First, we use interrelated tasks such as link prediction, and link classification to generate node embeddings that are added to the raw features to capture graph topological information, which is then used for training a supervised machine learning algorithm to detect fraudulent nodes.

Keywords: Graph Neural Network, Node Classification, Link Prediction, Link Classification

Chapter 1

Introduction

Fraud detection has always been a hot topic in the machine learning paradigm, as it helps to prevent economic losses and safeguards individuals and organizations from fraudulent activities. Fraud can take many distinct forms such as banking fraud, healthcare fraud, and social media fraud, and it can have notable negative impacts on society as a whole. Some of the ramifications of fraud include financial loss, reputation damage, and reduction in trust. According to Truman & Reuter (2004), the accurate dimension of money laundering transactions is undiscovered and unresolved, likely because of the financial ecosystem's inability to come up with tools that can accurately estimate the extent of money laundering from the ingoing and outgoing transactions in their accounts. A survey published by Hicks & Graycar (2019) at the United Nations Office on Drugs and Crime in 2011, estimated that the total amount of laundered money through financial systems roughly equals US $1.6 trillion. Additionally, the detection of fraudulent activities is an important task for many e-commerce, and social media websites, and not just limited to finance. In the work published by Chang et al. (2018), deceptive reviews on e-commerce websites degrade customer satisfaction because of their ability to mislead customers' opinions and decisions. Similarly, the work published by Akter et al. (2022) shows how detrimental comments by internet trolls on social media platforms hurt the general audience's mental health.

Moreover, in recent times, malpractitioners have become highly skilled in using cryptocurrencies for money laundering. Criminals leverage the pseudonymity of Cryptocurrencies like Bitcoins to hide their identities and transfer millions of dollars by using their illicit digital wallets. On the bright side, cryptocurrency transaction data are publicly available, meaning that open-source intelligence agencies and

law enforcement agencies have the required permission to conduct an analysis of the available data. Yet, the sheer volume of the data itself, and the untraceable Peer-to-Peer (P2P) cross-border nature of cryptocurrencies, as mentioned in Lo et al. (2022), make it far more challenging. According to Salminen et al. (2022), fake comments are posted by fake users who are either bots or individuals hired by businesses to write unequivocal reviews for their services or negative reviews for their competitors. Fake reviews are cumbersome to detect, existing methods include pattern matching, checking the reviewer's activity history, and account information, and then analyzing all this information for suspicious activity.

Fortunately, Graph Neural Networks have shown very promising results in fraudulent activity detection. One such work is proposed by Weber et al. (2019), where graph representation learning is leveraged in order to detect illicit transactions from the Elliptic dataset which is a graph-structured dataset for Bitcoin transactions. We intend to look into complementary tasks that will help us to learn better node representations to use as input for node classification, such as link prediction, link classification, or link regression. In link prediction, the goal is to infer the existence of edges between nodes in a graph, whereas in whereas in link classification a model is trained to predict the type of link in a graph. This relates to the idea of self-supervised learning based on graph reconstruction as discussed in Ju et al. (2022), since link prediction works by generating positive and negative views of the graph (i.e., by considering existing and non-existing edges, respectively), does not require additional labeled data and can be used to obtain better node representations. Similarly, link classification works by learning the relationships that exist between nodes in the graph, and using that information to classify different types of existing links in the graph. Using tasks such as link prediction and link classification as pretext task helps encode the structural information that the graph sustains.

Since the Elliptic dataset does not contain edge features or labels we look into the fraud Amazon dataset, and the fraud Yelp dataset mentioned in works proposed by Zhang et al. (2020) and Rayana & Akoglu (2015) respectively. The fraud Amazon and fraud Yelp datasets do not deal with fraud detection in cryptocurrency but contain information that can be used to identify fraudulent users and fake reviews respectively.

In this thesis, we propose to use multi-task learning, where complementary tasks such as link prediction and link classification are leveraged to generate node embeddings that are added to the raw features to increase our feature set, which is then used to train a supervised machine learning algorithm to detect fraudulent entities, such as illicit transactions for the Elliptic dataset, fraudulent users on the fraud Amazon and fake reviews on the fraud Yelp dataset.

In summary, our key contributions are:

- We use graph link prediction and link classification to learn the node embeddings without using any labels associated with each node. The node embeddings extracted from the link prediction and link classification tasks are then added to the raw node features to capture additional graph topological information.

- A comprehensive evaluation of fraud detection on the Elliptic dataset, fraud Amazon and fraud Yelp datasets, to check whether node classification results on imbalanced fraud datasets can be improved by augmenting original node features with node embeddings learned from other graph machine learning tasks.

Chapter 2

Related Work

In this section, we provide a review of related works. We focus on three major study topics related to our problem: (i) Graph Neural Networks, where we cover the most widely known neural network-based approaches for learning on graphs, (ii) Class Imbalance, in which we review many of the main methods for dealing with imbalanced classification scenarios, and (iii) Fraud detection on graphs, in which we review techniques for fraudulent activity detection.

2.1 Graph Neural Networks

According to Kipf & Welling (2016a), Graph Neural Networks (GNNs) have received major attention in recent years due to their novel potential to learn from data that models complex topological relationships between observations and therefore does not originally reside in a Euclidean space. GNNs leverage the concept of message passing, where each layer learns every node's embedding by aggregating its neighbor's information. In doing so, they allow relatively shallow networks to model complex topological information about the nodes on the graph with just a few layers.

Our work is mostly related to graph representation learning, i.e., learning node representations on euclidean space according to unsupervised or semi-supervised tasks. Hamilton et al. (2017) were amongst the first to propose an architecture for unsupervised learning in graphs called GraphSAGE. GraphSAGE's loss function is based on the concept of random walks, thus ensuring that the nearby nodes have more similar embeddings as compared to the distant nodes. The model also provides a generalization of the original GNN architecture, by allowing multiple

different aggregator functions in the message-passing mechanism. The Graph Attention network architecture proposed by Veličković et al. (2018) has also been successful, as it allows the network to learn which neighbors to prioritize while information aggregation, by utilizing masked self-attention.

There are also many recent approaches specifically for temporal graphs. Pareja et al. (2019), the creators of EvolveGCN, combine a Recurrent Neural Network with a GNN in order to process the temporal aspect of dynamic graphs. Their framework provides two different methods for employing these networks: either by utilizing the GNN weight matrix as the hidden state or as input of the dynamical system. Moreover, L. Hu et al. (2022) recently proposed Temporal Graph Attention (TGAT) which shows promising results when dealing with dynamic graphs. The TGAT network obtains every node's embedding by aggregating the hidden representations of its neighbors at a given time, and then applying the self-attention operation on these aggregated features, to output a time-aware representation of the target node.

The aforementioned approaches are designed for homogeneous graphs, however, there are many attempts to generalize GNNs for heterogeneous graphs too. Schlichtkrull et al. (2017) proposed RGCN which extends GNN to heterogeneous graphs, by learning on subgraphs created for each relation individually and then aggregating the information across all relations. More recent approaches, such as HGT by Z. Hu et al. (2020) combine elements of Transformer architecture Vaswani et al. (2017) with GNNs, to model heterogeneous temporal relations. These approaches generally focus on training the models directly on a downstream task. However, the Deep Graph Library Zheng et al. (2020) provides a general framework for learning knowledge graphs based on an unsupervised task, known as DGL-KE, that aims to learn node embeddings, by employing a similar strategy of learning on each relation independently and then aggregating the information into a single embedding.

2.2 Class imbalance

Class imbalance, as discussed by Longadge & Dongre (2013); Japkowicz & Stephen (2002), is inherent to real-world datasets, i.e, class instances are not equally distributed. When this difference becomes too large, it might become a problem because even robust machine learning methods might make inaccurate predictions

by always attributing new samples to the majority class. There are many strategies for dealing with this problem, but here we will be focusing on approaches that involve data manipulation. These data-level strategies seek to adjust class size through over- or under-sampling, i.e., making the majority classes smaller and the minority classes bigger. The naive form of over-sampling reduces the class imbalance by replicating existing samples, which may, however, cause overfitting as a side effect. Bowyer et al. (2011) proposed SMOTE a popular over-sampling algorithm, that interpolates observations belonging to minority classes with their nearest neighbors to create new synthetic observations. However, it does not consider topological information as it relies on traditional distance metrics to determine the nearest neighbors. On the other hand, GraphSMOTE proposed by Zhao et al. (2021), tries to solve this issue by adapting the SMOTE algorithm to better suit graph representation learning applications. It first extracts node embeddings by using a single-layer GNN and then applies SMOTE to balance the previously under-represented classes. It also creates an edge generator in order to connect these synthetic nodes to the network, by training a GNN-based classifier that reconstructs the original adjacency matrix of the network, and then proceeds to train a final classifier for node classification on the downstream task. GraphSMOTE is one of the most prominent approaches to over-sampling in the context of graph representation learning, achieving state-of-the-art results on imbalanced datasets, and in Chapter 4, we assess its applicability to our work

2.3 Fraud detection on graphs

Detecting money laundering activity on graph-structured data can be seen as a special case of detecting illicit activity on transaction networks. Starnini et al. (2021) provide an in-depth overview of GNN methods in finance. All of these are essentially related to imbalanced scenarios, where the illicit activity is carried out by a smaller minority. The imbalance ratio is determined by the rarity of these activities and must be taken into account when constructing models for these problems.

Many of the proposed models focus on financial fraud detection. Alarab & Prakoonwit (2022) provide a supervised model for classifying fraudulent activity in heterogeneous graphs. While they do not focus on class imbalance, they do provide a framework for processing and aggregating information of nodes, by creating an

attention-based graph convolution layer that considers the heterogeneous nature of the graph. There are also successful semi-supervised approaches for fraud detection, such as SemiGNN, proposed by Wang et al. (2019). SemiGNN provides an attention-based method, by defining an unsupervised loss function based on the concept of DeepWalks as discussed in Perozzi et al. (2014). Camouflage-Resistant GNN (CARE-GNN) proposed by Dou et al. (2020), is a graph neural network architecture that allows to capture the complex relationships between different image features and the presence of camouflage. However, all of the aforementioned models do not directly deal with the imbalance problem that is prominent in our data. Interesteingly, GraphSMOTE proposed by Zhao et al. (2021) is an oversampling technique that addresses the issue of class imbalance, but it can be computationally expensive, especially for large graph datasets. GraphSMOTE involves interpolating new nodes and building a new bigger graph that incorporates all the newly generated nodes, which makes it time-consuming and memory-intensive.

Chapter 3

Technical Background

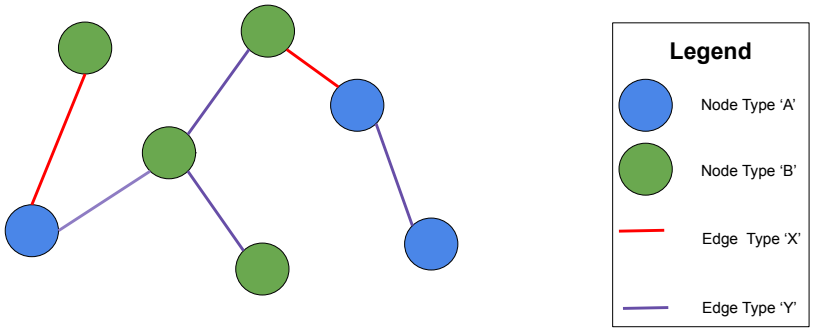

Figure 3.1: Heterogenous Graph with multiple edge types and node types

A graph is an arrangement comprising a collection of nodes, where certain pairs of nodes exhibit some form of association. Mathematically it can be represented as $G = (V, E)$, where V is the set of nodes and E depicts the set of edges. Graphs can be heterogeneous or homogenous. In a heterogeneous graph the nodes and/or edges can belong to different types, while in a homogeneous graph, all the nodes and edges belong to the same type. Every node and edge can have a set of features associated with it, which contains information particular to that node or edge. In this thesis, we explore machine learning tasks on graphs, such as node classification, link prediction, and link classification. The following subsection elaborates on the formal definitions for each of the different machine learning tasks on graphs, and the logic behind graph convolution network.

3.1 Node classification

Node classification is a task where the aim to predict the class label of each existing nodes in a graph. Mathematically, given a graph $G = (V, E)$ where V is the set of nodes and E is the set of edges. G also has a feature matrix X and label vector Y, thus the goal of node classification is to learn a mapping function $f : V \rightarrow Y$ that maps the features of each node to its corresponding class label. It is done by minimizing a loss function

$$L = -\sum_{c=1}^{K} y_c \log(p_c), \tag{3.1}$$

where K is the number of classes for nodes, y is the binary indicator (0 or 1) determining whether the class label c is the correct class label for that node, and p is predicted class probability for that node. Node classification is possible only for heterogenous graphs.

3.2 Link prediction

Link prediction aims to predict the presence of links (i.e., edges) between nodes in a graph. It has many applications in various domains, such as social networks, and recommendation systems. Given a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges, the goal of link prediction is to predict the presence or absence of a link between a pair of nodes $(u, v) \in E$ such that $u \neq v$ and $(u, v) \notin E$. This is done by minimizing a loss function

$$L = -(y \log(p) + (1 - y) \log(1 - p)), \tag{3.2}$$

where y is the binary indicator (0 or 1) determining whether the link exists or not, and p is the predicted link probabilities to the true link labels. Link prediction is possible for both homogenous and heterogenous graphs.

3.3 Link classification

Link classification aims to predict the class label of a link (an edge) between two nodes in a graph. Given a graph $G = (V, E)$ with feature matrix X and label vector

Y, the goal of link classification is to learn a function $f : V \times V \to Y$ that maps the features of each link $(u, v) \in E$ to its corresponding class label. It is done minimizing a loss function

$$L = -\sum_{c=1}^{C} y_c \log(p_c), \tag{3.3}$$

where C is the number of classes for link, y is the binary indicator (0 or 1) determining whether the class label c is the correct class label for that link, and p is predicted class probability for that link. Link classification is possible only for heterogenous graphs.

3.4 Graph Convolutional Networks

Graph Convolutional Networks (GCNs) proposed by Kipf & Welling (2017) belong to the class of neural networks that can handle graph-structured datasets. GCN extends the concepts of convolutional neural networks (CNN) to non-Euclidean domains, such as graphs, by defining a convolution operation on graphs. GCNs learn representations of nodes in a graph by propagating information through the graph structure, capturing both local and global dependencies.

Let G be a graph $G = (V, E)$ where V is the set of nodes and E is the set of edges. Let X be the feature matrix of size $N \times D$, where N is the number of nodes and D is the number of features for each node. Edge relationships are defined by adjacency matrix A of size $N \times N$, where $A_{ij} = 1$ if an edge exists between node i and node j, else $A_{ij} = 0$.

The convolution operation in GCN on graph G can be decomposed into two major steps: message passing, and parameter passing. In message passing each node aggregates the features of its neighbors, and in parameter sharing all nodes share the same weights. Formally the convolution operation is defined as

$$H = g(A, X, W) \tag{3.4}$$

where, H is the output feature matrix of size $N \times F$, where F is the number of output features for each node, W is the weight matrix of size $D \times F$, and g is an activation function.

The message-passing step is defined by

$$M = AX \tag{3.5}$$

where M is the message matrix of size $N \times D$, and each row i corresponds to the aggregation of the features of the neighbors of node i.

The message passing step in a GCN can be improved by adding self-loops and using the graph Laplacian operator. Self-loops are added to the adjacency matrix A by $\hat{A} = A + I$. The graph Laplacian operation is defined as $L = D^{-1/2}\hat{A}D^{-1/2}$, where D is the diagonal degree matrix of $\hat{A}$, with $D_{ii} = \sum_j \hat{A}_{ij}$.

Thus the message passing step in a GCN can then be defined as:

$$M = D^{-1/2}\hat{A}D^{-1/2}X \tag{3.6}$$

where M is the message matrix of size $N \times F$.

The parameter sharing step can be defined as

$$H = g(MW) \tag{3.7}$$

where, H is the output feature matrix of size $N \times F$, and each row i corresponds to the activation of the aggregated features of node i. The weight matrix W is learned through backpropagation during training, thus allowing the GCN to adapt to the specific task at hand.

GCN can be applied to various graph-related tasks, including node classification, link prediction, and link classification. In node classification, the basic idea is to first define a graph Laplacian matrix L, which is a square matrix of size $n \times n$, where n is the number of nodes in the graph. The diagonal element L_{ii} of the Laplacian matrix equals the degree of node i, and L_{ij} such that $i \neq j$ equals -1 if there is an edge between nodes i and j, and 0 otherwise. The convolution operation then involves multiplying the input feature matrix X by L and a weight matrix W and then applying a non-linear activation function such as ReLU or sigmoid. The output of the graph convolutional layer is a feature matrix H, which incorporates information from both the input features and the graph structure. This process is repeated for as many layers of GCN, with the final output used for node classification.

In link prediction, the goal is to predict the presence or absence of a link between two nodes in a graph, by computing a similarity score between pairs of nodes. The

basic idea is to generate positive and negative views of the graph (i.e., by considering existing and non-existing edges, respectively) A graph Laplacian matrix L, similar to the one discussed in node classification is defined which incorporates both the adjacency matrix and the node and edge features. The graph convolution operation then involves multiplying the input feature matrix X by L and a weight matrix W and applying a non-linear activation function such as ReLU or sigmoid. The output of the convolutional layer is a new feature matrix H, which incorporates information from both the input features and the graph structure. This process is repeated for multiple layers of graph convolution, with the final output used to compute the similarity score between pairs of nodes.

The goal for link classification, is to predict the type or category of a link between two nodes in a graph, based on the graph structure and other node or edge features. It is similar to link prediction but involves an additional step of applying a classification layer to the final output of the graph convolutional layers. The final classification layer can be a fully connected layer or a softmax function, depending on the specific task and the number of classes of links to be predicted. The output of the last classification layer is a probability distribution over the possible classes of links in the graph, which is used to make the final prediction.

3.5 Multi-task learning

In machine learning, multi-task learning is a technique where a single model is trained to perform multiple complementary tasks simultaneously. Instead of training a different model for each task, a multi-task learning model learns to perform multiple tasks by sharing information between them. It allows a single model to be trained to perform multiple tasks, reducing the amount of training data needed and improving performance on all tasks. The key idea behind multitask learning is that complementary tasks can benefit from each other's data and can share learned representations. For example, a model trained to predict the existence of links in a graph, or link types for a graph may learn to recognize common features such as the graph's structural pattern that are useful for both tasks.

In multi-task learning, we have a set of tasks $\mathcal{T} = T_1, T_2, ..., T_n$, where each task T_i is associated with a dataset $\mathcal{D}i = (x_{i,1}, y_{i,1}), (x_{i,2}, y_{i,2}), ..., (x_{i,m_i}, y_{i,m_i})$, where $x_{i,j}$ is an input and $y_{i,j}$ is a corresponding output for task T_i. And our goal is to learn a

single model that can perform well on all the tasks. In multitask learning, we learn a single set of parameters W that is shared across all complementary tasks, instead of learning a separate set of parameters W_i for each task T_i.

The cost function for multitask learning can be mathematically written as

$$\mathcal{L}(W) = \sum_{i=1}^{n} \sum_{j=1}^{m_i} \mathcal{L}i(f(xi, j; W), y_{i,j}) + \lambda\Omega(W), \tag{3.8}$$

where $\mathcal{L}i$ is the loss function for task T_i, $f(xi, j; W)$ is the output of the model for input $x_{i,j}$ with parameters W, λ is a regularization parameter, and $\Omega(W)$ is a regularization term.

The first term in the cost function represents the specific task-related loss, which measures the difference between the predicted output and the true output for each task. The second term is a regularization term (such as L1 or L2) that encourages the parameters to be shared across tasks without becoming prone to overfitting. The minimization of the cost function allows us to learn a set of parameters W that perform well on all complementary tasks. During the training process, the model updates the shared parameters on the basis of the gradients computed from the task-specific losses. This allows information sharing between tasks thus the model can learn to generalize better and improve performance on all tasks.

Chapter 4

Methodology

Our goal is to perform fraudulent node classification for graph datasets. In the approach discussed by Weber et al. (2019), they implemented node classification using a variant of GCN that inserts a residual connection between the intermediate embedding and the input node features, to generate node embeddings, which are in turn concatenated with raw node features to train a Random Forest. However, one of the major problems in fraud detection is class imbalance as discussed in Section 2.2. In order to overcome that, we have leveraged multi-task learning, where complementary graph representation learnings such as link prediction and link classification are used to generate node embeddings to enrichen the raw node feature set before the final classification task.

This section is divided into three subsections: (i) Datasets, where we briefly describe the three datasets that we have used for fraud detection, (ii) Proposed method, which discusses our approach, and (iii) Experimental setup, in which we elaborate on our implementation of graph representation learning

Dataset	# of Nodes	# of Edges	# of Node types	# of Edge types
Elliptic	203,769	234,355	3	1
Fraud Amazon	11,944	9,557,648	2	3
Fraud Yelp	45,954	8,051,348	2	3

Table 4.1: About dataset

4.1 Datasets

In this thesis, we have used the graph datasets that deal with fraud detection. We have used the Elliptic dataset, the fraud Amazon dataset, and the fraud Yelp dataset. We initially had started with the Elliptic dataset, a labeled graph-structured dataset of bitcoin transactions, to compare our results to those of Weber et al. (2019). As we wanted to explore other complementary tasks such as link prediction and link classification to get better embeddings that can enrich the raw feature set for the final classification task, we further looked for datasets suitable for link classification too, i.e, those that have heterogeneous edge types. The datasets that also deal with fraudster detection are the fraud Amazon dataset and the fraud Yelp dataset, as discussed by Zhang et al. (2020) and the Rayana & Akoglu (2015) respectively. Both of these two datasets have multiple edge types, thus allowing link classification. The fraud Amazon and fraud Yelp datasets include users' product reviews on Amazon and Yelp, respectively. Table 4, summarizes all the datasets that have been used.

4.1.1 Elliptic dataset

The Elliptic dataset is a labeled graph-structured dataset of bitcoin transactions, where nodes are transactions and directed edges depict the transaction flow (i.e., the flow of bitcoins from one transaction to another). The dataset has a total of 203,769 nodes and 234,355 edges. Nodes in the graph are classified into two major categories- 21% (42,019) of the nodes are labeled licit, 2% (4,545) as illicit, and the rest are unknown. Licit transactions comprise transactions related to currency exchanges, crypto mining, and other legal transactions, while illicit transactions comprise money transfers initiated by scams, malware attacks, and terrorist organizations. There are 166 node features. The first 94 called Local Features, represent the transaction's local information such as the transaction fee, amount of average bitcoins received, etc. The last 72 features called Aggregated Features, represent statistics obtained after aggregating information from neighbors that are one-hop backward or forward from that particular transaction node, such as correlation coefficients, and variance of the neighbor nodes. The dataset is grouped into 49 different timesteps that are evenly spaced over a 3 hours gap within an interval of 2 weeks.

4.1.2 Fraud Amazon dataset

The fraud Amazon dataset has 11,944 nodes depicting users classified over two main categories - benign and fraudulent. Benign users (7,818) are users with more than 80% helpful votes on their reviews, fraudulent users (821) are users with reviews that have lesser than 20% helpful votes, and the rest of the users (3,305) are unknown. The edges in the graph are classified into three major categories, U-P-U (351,216) connecting users that reviewed at least one similar product, U-S-U (7,132,958) connecting users that have at least one similar star rating within one week, and U-V-U (2,073,474) connecting users having 5% text similarities on their reviews measured using the TF-IDF statistic. The Fraud Amazon dataset has a total of 25 node features.

4.1.3 Fraud Yelp dataset

The fraud Yelp dataset has 45,954 nodes depicting reviews classified over two main categories- spam (6,677)and legitimate (39,277). The edges in the graph are classified into three major categories, R-U-R (98,630) connecting reviews posted by the same user, R-S-R (1,147,232) connecting reviews under the same product with the same star rating (1-5 stars), and R-T-R (6,805,486) connecting two reviews under the same product posted in the same month. The fraud Yelp dataset has a total of 32 node features.

4.2 Proposed Method

We proposed a multi-task learning framework, where we perform complementary graph representation learning tasks such as link prediction and link classification, to generate node embeddings. The node embeddings generated by the complementary tasks are then concatenated with the raw node features to train a random forest classifier. While training for the complementary tasks, we did not consider the node labels associated with nodes to avoid data leakage.

In our experiments, we use the GNN known as the Graph Convolutional Network (GCN), originally proposed by Kipf & Welling (2016b). GCN bags the notion of convolution from the convolutional neural network (CNN) and convolves the graph directly by using the graph connectivity structure as the filter to perform

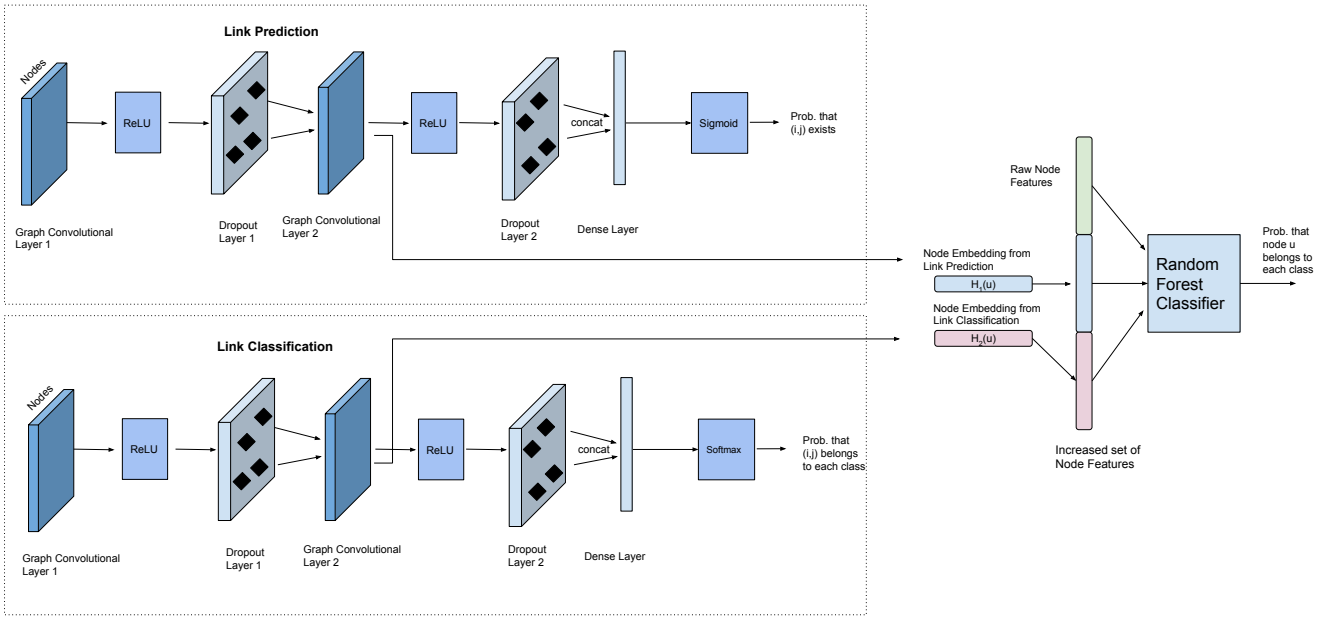

Figure 4.1: Proposed approach.

the neighborhood aggregation. Let $G = (V, E)$ be a graph, where V is the set of nodes and E depicts the set of edges. In a GCN with L layers, The l-th layer of the GCN model takes the adjacency matrix A of the graph G and the node embedding matrix $H^{(l)}$ as the input and employs a weight matrix $W^{(l)}$ to generate the node embedding matrix $H^{(l+1)}$ as output. Mathematically, it can be summarized as

$$\mathbf{H}^{(l+1)} = \mathbf{g}(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}), \tag{4.1}$$

where we get after the normalization of A defined as

$$\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}, \tag{4.2}$$

$$\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}, \tag{4.3}$$

$$\tilde{\mathbf{D}} = \mathbf{diag}([\Sigma_j \tilde{\mathbf{A}}_{ij}]_i), \tag{4.4}$$

In the case of link prediction or link classification, g is the activation function. Generally, ReLU activation function is used for all layers except the last. In link prediction the last layer has a sigmoid activation function, but in link classification the last layer has softmax activation function. The raw features X serve as the initial embedding matrix $H(0)$.

Our implementation of the GCN architecture has two layers, each with a certain dropout rate to avoid overfitting. The GCN takes labeled pairs of nodes corresponding to possible transaction links and outputs the node embeddings. These embeddings are then passed to a link classification layer, which performs element-wise multiplication (referred as $\odot$ operator in equation 4.5) to generate the embeddings for the links. These link embeddings are then fed to a dense link classification layer to generate the final predictions. The prediction can be expressed as

$$\mathbf{h}_{ij} = \mathbf{W}_1(\mathbf{h}_i^{(L)} \odot \mathbf{h}_j^{(L)}) + \mathbf{b}_1, \tag{4.5}$$

$$\mathbf{p_{ij}} = g(\mathbf{W}_2^\top \mathbf{h}_{ij} + \mathbf{b_2}), \tag{4.6}$$

The model is trained by minimizing the loss function as discussed in Section 3.2 for link prediction and Section 3.3 for link classification.

4.3 Experimental setup

Hyperparameter	Link Prediction
GCN shape	(100,16)
Number of epochs	500
Optimizer	Adam
Dropout rate	30%
Learning rate	0.01

Table 4.2: Hyperparameters for the Elliptic dataset

Hyperparameter	Link Prediction	Link Classification
GCN shape	(20,10)	(20,10)
Number of epochs	20	20
Optimizer	Adam	Adam
Dropout rate	10%	10%
Learning rate	0.1	0.1

Table 4.3: Hyperparameters for fraud Amazon and fraud Yelp datasets

This subsection discusses our implementation of graph representation learning on three different datasets, the Elliptic dataset, the fraud Amazon dataset, and the fraud Yelp dataset.

Table 4.2 summarizes the hyperparameters that have been used while exploring the Elliptic dataset. We implemented a two-layer GCN model for link prediction on the Elliptic dataset. The first layer has 100 hidden units, and the second layer has 16 hidden units. The Adam optimizer with a learning rate of 0.01 is then to update the GCN parameters by using mini-batches of training links fed to the model for a total of 500 epochs. We have used a dropout rate of 30% for each layer to avoid overfitting. The embeddings from the second layer of the GCN model used for link prediction are then concatenated to the raw feature set. This combined set of features is then used to train a Random Forest Classifier for the final Illicit transaction classification task.

Table 4.3 summarizes the hyperparameters for the fraud Amazon and fraud Yelp datasets. Both the fraud Amazon and fraud Yelp datasets have multiple edge types, thus allowing link classification along with link prediction. We implemented two separate two-layer GCN models for link prediction and link classification. Each of the GCN models has 20 hidden units for the first layer, and 10 for the second layer, and is trained using the Adam optimizer. Both models use 0.1 as the learning rate and are trained using mini-batches of training links for a total of 20 epochs. The embeddings generated by the second layer of each GCN model are then added to the raw node feature set to train a Random Forest Classifier for the final classification task, i.e., fraudulent user detection for the fraud Amazon dataset and fake review detection for the fraud Yelp dataset.

Chapter 5

Results

In this chapter, we describe the results from running our experiments on the Elliptic dataset, fraud Amazon dataset, and fraud Yelp dataset. The metrics that we have used for comparing the results are Precision, Recall, F1 score, Micro F1 score, Area Under the ROC Curve (AUC), and Area Under the Precision-Recall Curve (AUPR). Formally, the metrics can be summarized as:

$$Precision = \frac{TP}{(TP + FP)} \tag{5.1}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{5.2}$$

$$F1\ score = \frac{TP}{(TP + \frac{1}{2}(FP + FN))} \tag{5.3}$$

where, in a two-class classification problem, TP is the number of True Positives and TN is the number of True Negatives. Similarly, FP and FN are the number of False Positives and False Negatives respectively. Micro F1 is calculated by taking the weighted average of F1 score for each of the existing class label.

The formula for generating the Area Under the ROC Curve (AUC) is:

$$AUC = \int_0^1 ROC(x)dx, \tag{5.4}$$

where $ROC(x)$ is the Receiver Operating Characteristic (ROC) curve at a specific threshold x, and the integral is taken over the entire range $[0, 1]$ of possible threshold

Method	Feature				Illicit			Micro-F1
	LF	AF	NE	LP	Precision	Recall	F1 score	
Skip GCN		✓	✓		0.971	0.675	0.796	0.978
	✓		✓		0.878	0.668	0.759	0.973
GCN (proposed)		✓		✓	**0.992**	**0.887**	**0.937**	**0.988**
	✓			✓	**0.988**	**0.854**	**0.916**	**0.984**

Table 5.1: Illicit transaction classification results. Types of features: LF = Local features; AF = Aggregated features; LP = Node embeddings from Link Prediction; NE = Node embeddings from Node Classification.

values. The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) for different threshold values, where TPR and FPR are defined as follows

$$TPR = TP/(TP + FN), \tag{5.5}$$

$$FPR = FP/(FP + TN) \tag{5.6}$$

The formula for generating the Area Under the Precision-Recall Curve (AUPR) is:

$$AUPR = \int_0^1 PR(p)dp, \tag{5.7}$$

where $PR(p)$ is the Precision-Recall (PR) curve at a specific threshold p, and the integral is taken over the entire range $[0, 1]$ of possible threshold values. The PR curve plots precision (P) against recall (R) for different threshold values, where precision and recall are defined in equation 5.1, and equation 5.2:

Table 5.1 shows our results from the experiment on the Elliptic dataset. Skip-GCN shows the result for Illicit classification using embeddings extracted from node classification. GCN shows results where link prediction was used for learning the node embeddings. Our approach performs better than the method proposed by Weber et al. (2019). The comparison is based on the Random Forest(RF) classifier trained using the node embeddings extracted from respective GNN models along with the raw features. While evaluating our model, we get higher precision, recall, and F1 scores. The goal of our experiment was to show that the embeddings learned from complementary tasks such as link prediction was useful for getting a better representation of the transactions thus, the overall performance of the model improves.

Method	Feature			Fraud					Micro-F1
	NF	LP	LC	Precision	Recall	F1 score	AUC	AUPR	
CARE-GNN	✓			–	**0.854**	–	0.884	–	–
GCN (proposed)	✓			0.927	0.776	0.845	0.883	0.855	0.980
	✓	✓		0.946	**0.784**	**0.860**	**0.885**	**0.861**	0.979
	✓		✓	**0.949**	0.764	0.850	0.884	0.859	**0.981**
	✓	✓	✓	**0.959**	0.776	**0.858**	**0.887**	**0.875**	**0.982**

Table 5.2: Fraudulent users classification results on fraud Amazon dataset. Types of features: NF = Raw node features; LP = Node embeddings from Link Prediction; LC = Node embeddings from Link Classification.

Method	Feature			Illicit					Micro-F1
	NF	LP	LC	Precision	Recall	F1 score	AUC	AUPR	
CARE-GNN	✓			–	**0.666**	–	0.685	–	–
GCN (proposed)	✓			0.882	0.432	**0.582**	**0.710**	0.698	**0.909**
	✓	✓		**0.895**	0.432	**0.581**	0.707	**0.699**	0.979
	✓		✓	0.882	**0.434**	0.580	0.704	0.697	0.980
	✓	✓	✓	**0.898**	0.425	0.577	**0.708**	**0.704**	**0.909**

Table 5.3: Fraudulent reviews classification results on fraud Yelp dataset. Types of features: NF = Raw node features; LP = Node embeddings from Link Prediction; LC = Node embeddings from Link Classification.

Table 5.2 shows our results from the experiment on the fraud Amazon dataset. First GCN has been used for link prediction followed by link classification to get node embeddings to increase the total raw feature set. A Random Forest classifier is trained on the enriched feature set for the final fraudulent user classification task.

Table 5.3 shows our results from the experiment on the fraud Yelp dataset. First GCN has been used for link prediction followed by link classification to get node embeddings to increase the total raw feature set. A Random Forest classifier is trained on the enriched feature set for the final fraudulent user classification task.

Our approach of using complementary tasks for graph representation learning such as link prediction and link classification using GCN improves the performance by capturing the graph topological information. We experienced the precision-recall trade off. The precision-recall trade off arises because increasing the precision often requires increasing the model's threshold for making a positive prediction, which can lead to a decrease in recall. Similarly, increasing the recall often requires lowering the model's threshold for making a positive prediction, which can lead to

a decrease in precision. For our work, precision is important because it measures the model's ability to correctly identify frauds, without making too many false positive predictions. In Fraud detection, false detection of frauds could lead to unnecessary overhead for the organization to investigate cases that are not a fraud. Therefore, it is important to have a high precision than a high recall for such applications.

Chapter 6

Conclusion

The thesis explores the challenges of fraud detection, such as illicit transactions in the financial ecosystem and forged review detection on popular e-commerce websites. Frauds serve a negative impact as it hurts the economy, trust, and reputation. Fraud detection has become an important topic in machine learning as it helps prevent economic losses and safeguards individuals and organizations from malicious activities. In this thesis, we inspect heterogeneous graph datasets that deal with fraud detection such as the Elliptic dataset, the fraud Amazon dataset, and the fraud Yelp dataset. The Elliptic dataset does not have multiple edge types but the fraud Amazon and fraud Yelp datasets contain multiple edge types. Our aim is to classify nodes of the graph as fraudulent or non-fraudulent.

One of the major obstacles in classification problems, especially in fraud detection, is the class imbalance ratio. Class imbalance is a phenomenon that occurs when the number of examples in each class of a dataset is not evenly distributed, for example, the ratio between the number of illicit transactions and that of licit transactions in a fraud detection problem is very small. As a way to tackle that, we explore complementary graph representation tasks such as link prediction and link classification, to generate node embeddings that enrichen our raw feature set before the final node classification task. In this thesis, both link prediction and link classification have been implemented by using Graph Convolutional Networks (GCN), which extends the concepts of convolutional neural networks (CNN) to non-Euclidean domains, such as graphs, by defining a convolution operation on graphs. GCNs learn representations of nodes in a graph by propagating information through the graph structure, capturing both local and global dependencies in two

major steps: message passing and parameter sharing. We also discuss the other Graph Neural Networks that have shown very promising results in fraud detection.

We have compared our results to other baseline methods such as Skip-GCN and CAmouflage-REsistant GNN (CARE-GNN) using popular metrics such as precision, recall, F1 score, AUC, AUPR, and Micro F1 score. In evaluating our results we experienced the precision-recall trade off. The precision-recall trade off arises because increasing the precision often requires increasing the model's threshold for making a positive prediction, which can lead to a decrease in recall. Similarly, increasing the recall often requires lowering the model's threshold for making a positive prediction, which can lead to a decrease in precision. For our work, precision is important because it measures the model's ability to correctly identify frauds, without making too many false positive predictions. In Fraud detection, false detection of frauds could lead to unnecessary overhead for the organization to investigate cases that are not a fraud. Therefore, it is important to have a high precision than a high recall for such applications. Our approach outperformed the baselines in metrics such as micro-F1 and AUC score which summarizes the overall performance of a model across all possible threshold values.

We also wanted to explore graph neural networks that are capable of handling class imbalance such as GraphSMOTE. GraphSMOTE uses an oversampling technique that addresses the issue of class imbalance, but it can be computationally expensive, especially for large graph datasets. The oversampling method involves interpolating new nodes and building a new bigger graph that incorporates all the newly generated nodes, which makes it time-consuming and memory-intensive. New approaches that can make GraphSMOTE scale better can be a promising direction for future work.

References

Akter, M. S., Shahriar, H., Ahmed, N., & Cuzzocrea, A. (2022, dec). Deep learning approach for classifying the aggressive comments on social media: Machine translated data vs real life data. In *2022 IEEE international conference on big data (big data)*. IEEE. Retrieved from https://doi.org/10.1109%2Fbigdata55660.2022.10020249 doi: 10.1109/bigdata55660.2022.10020249

Alarab, I., & Prakoonwit, S. (2022, Jun 16). Graph-based lstm for anti-money laundering: Experimenting temporal graph convolutional network with bitcoin data. *Neural Processing Letters*. Retrieved from https://doi.org/10.1007/s11063-022-10904-8 doi: 10.1007/s11063-022-10904-8

Bowyer, K. W., Chawla, N. V., Hall, L. O., & Kegelmeyer, W. P. (2011). SMOTE: synthetic minority over-sampling technique. *CoRR, abs/1106.1813*. Retrieved from http://arxiv.org/abs/1106.1813

Chang, S., Zhenzhong, X., & Xuan, G. (2018). *Fake comment detection based on sentiment analysis.*

Dou, Y., Liu, Z., Sun, L., Deng, Y., Peng, H., & Yu, P. S. (2020, oct). Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM international conference on information &amp knowledge management*. ACM. Retrieved from https://doi.org/10.1145%2F3340531.3411903 doi: 10.1145/3340531.3411903

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. *CoRR, abs/1706.02216*. Retrieved from http://arxiv.org/abs/1706.02216

Hicks, D. C., & Graycar, A. (2019). Money laundering. In M. Natarajan (Ed.), *International and transnational crime and justice* (2nd ed., p. 80–85). Cambridge University Press. doi: 10.1017/9781108597296.013

Hu, L., Liu, S., & Feng, W. (2022). *Spatial temporal graph attention network for skeleton-based action recognition.* arXiv. Retrieved from https://arxiv.org/abs/2208.08599 doi: 10.48550/ARXIV.2208.08599

Hu, Z., Dong, Y., Wang, K., & Sun, Y. (2020). Heterogeneous graph transformer. *CoRR, abs/2003.01332*. Retrieved from https://arxiv.org/abs/2003.01332

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 429–449.

Ju, M., Zhao, T., Wen, Q., Yu, W., Shah, N., Ye, Y., & Zhang, C. (2022). *Multi-task self-supervised graph neural networks enable stronger task generalization.* arXiv. Retrieved from https://arxiv.org/abs/2210.02016 doi: 10.48550/ARXIV.2210.02016

Kipf, T. N., & Welling, M. (2016a). Semi-supervised classification with graph convolutional networks. *CoRR, abs/1609.02907*. Retrieved from http://arxiv.org/abs/1609.02907

Kipf, T. N., & Welling, M. (2016b). *Semi-supervised classification with graph convolutional networks.* arXiv. Retrieved from https://arxiv.org/abs/1609.02907 doi: 10.48550/ARXIV.1609.02907

Kipf, T. N., & Welling, M. (2017). *Semi-supervised classification with graph convolutional networks.*

Lo, W. W., Kulatilleke, G. K., Sarhan, M., Layeghy, S., & Portmann, M. (2022). *Inspection-l: Self-supervised gnn node embeddings for money laundering detection in bitcoin.* arXiv. Retrieved from https://arxiv.org/abs/2203.10465 doi: 10.48550/ARXIV.2203.10465

Longadge, R., & Dongre, S. (2013). Class imbalance problem in data mining review. *CoRR, abs/1305.1707*. Retrieved from http://arxiv.org/abs/1305.1707

Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., ... Leiserson, C. E. (2019). Evolvegcn: Evolving graph convolutional networks for dynamic graphs. *CoRR, abs/1902.10191*. Retrieved from http://arxiv.org/abs/1902.10191

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014, aug). DeepWalk. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining.* ACM. Retrieved from https://doi.org/10.1145%2F2623330.2623732 doi: 10.1145/2623330.2623732

Rayana, S., & Akoglu, L. (2015). Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining* (p. 985–994). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2783258.2783370 doi: 10.1145/2783258.2783370

Salminen, J., Kandpal, C., Kamel, A. M., gyo Jung, S., & Jansen, B. J. (2022). Creating and detecting fake reviews of online products. *Journal of Retailing and Consumer Services*, *64*, 102771. Retrieved from https://www.sciencedirect.com/science/article/pii/S0969698921003374 doi: https://doi.org/10.1016/j.jretconser.2021.102771

Schlichtkrull, M., Kipf, T. N., Bloem, P., Berg, R. v. d., Titov, I., & Welling, M. (2017). *Modeling relational data with graph convolutional networks.* arXiv. Retrieved from https://arxiv.org/abs/1703.06103 doi: 10.48550/ARXIV.1703.06103

Starnini, M., Tsourakakis, C. E., Zamanipour, M., Panisson, A., Allasia, W., Fornasiero, M., ... Moncalvo, D. (2021). Smurf-based anti-money laundering in time-evolving transaction networks. In *Machine learning and knowledge discovery in databases. applied data science track* (pp. 171–186). Springer International Publishing. Retrieved from https://doi.org/10.1007%2F978-3-030-86514-6_11 doi: 10.1007/978-3-030-86514-6_11

Truman, E., & Reuter, P. (2004). *Chasing dirty money: The fight against anti-money laundering* (Vol. 84). doi: 10.2307/20034366

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *CoRR*, *abs/1706.03762*. Retrieved from http://arxiv.org/abs/1706.03762

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). *Graph attention networks.*

Wang, D., Qi, Y., Lin, J., Cui, P., Jia, Q., Wang, Z., ... Yang, S. (2019, nov). A semi-supervised graph attentive network for financial fraud detection. IEEE. Retrieved from https://doi.org/10.1109%2Ficdm.2019.00070 doi: 10.1109/icdm.2019.00070

Weber, M., Domeniconi, G., Chen, J., Weidele, D. K. I., Bellei, C., Robinson, T., & Leiserson, C. E. (2019). *Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics.* arXiv. Retrieved from https://arxiv.org/abs/1908.02591 doi: 10.48550/ARXIV.1908.02591

Zhang, S., Yin, H., Chen, T., Hung, Q. V. N., Huang, Z., & Cui, L. (2020). *Gcn-based user representation learning for unifying robust recommendation and fraudster detection.* arXiv. Retrieved from https://arxiv.org/abs/2005.10150 doi: 10.48550/ARXIV.2005.10150

Zhao, T., Zhang, X., & Wang, S. (2021). Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proceedings of the 14th acm international conference on web search and data mining* (pp. 833–841).

Zheng, D., Song, X., Ma, C., Tan, Z., Ye, Z., Dong, J., … Karypis, G. (2020). Dgl-ke: Training knowledge graph embeddings at scale. In *Proceedings of the 43rd international acm sigir conference on research and development in information retrieval* (p. 739–748). New York, NY, USA: Association for Computing Machinery.