UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Ronald Davi Rodrigues Pereira

Detecção Automática de Comportamentos Fraudulentos em Redes utilizando Aprendizado em Grafos

Belo Horizonte
2021

Ronald Davi Rodrigues Pereira

Detecção Automática de Comportamentos Fraudulentos em Redes utilizando Aprendizado em Grafos

Versão final

Dissertação apresentada ao Programa de Pós--Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: Fabricio Murai Ferreira

Belo Horizonte

2021

Pereira, Ronald Davi Rodrigues.

P436a Automatic detection of fraudulent behavior in networks using graph learning [manuscrito] / Ronald Davi Rodrigues Pereira — 2021.
54 f. il.

Orientador: Fabricio Murai Ferreira.
Dissertação (mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação
Referências: f.50-54

1. Computação – Teses. 2. Redes neurais (Computação) – Teses. 3. Detecção de fraude – Teses. I. Ferreira, Fabricio Murai. II. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação. III.Título.

CDU 519.6*32 (043)

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa
CRB 6ª Região nº 1510

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Automatic Detection of Fraudulent Behavior in Networks using Graph Learning

RONALD DAVI RODRIGUES PEREIRA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

[illegible]

Prof. Fabrício Murai Ferreira - Orientador
Departamento de Ciência da Computação - UFMG

[illegible]

Prof. Pedro Olmo Stancioli Vaz de Melo
Departamento de Ciência da Computação - UFMG

[illegible]

Prof. Daniel Sadoc Menasche
Departamento de Ciência da Computação - UFRJ

Belo Horizonte, 25 de Outubro de 2021.

Acknowledgments

Over the past two years, I have had many excellent experiences in my Masters, for which I am very grateful. First, I would like to thank my parents, Miriam and Arnaldo, for always investing in me so that I could get where I am today. I will never be able to thank them for everything they did and still do for me.

Second, I would like to thank my girlfriend, Marcella, for always putting up with and supporting me through the difficult times in my life. There were many times I talked to her about extremely complex things to resolve and, as much as she would not know how to help me resolve it, she gave me the emotional support so that I could overcome these barriers.

Thirdly, I would like to thank my advisor, Fabricio Murai, for having helped me so much in these two years of my master's degree. Many teachings he gave me go far beyond the academic side and will remain for the rest of my life.

Fourth, I would like to thank my friends and co-workers at Hekima and, later, at iFood, who made it possible for me to make this investment in academic qualification without any implications for my work, especially Thiago Cardoso and João Martins.

Last but not least, I would like to thank my dog, Ralph, for always managing to cheer me up in my times of greatest need with his unconditional love and unparalleled care.

I feel honored to have all these people (and animals) around me and the others that I consider but haven't mentioned. I would like to tell everyone that I greatly appreciate all the support you have given me during these two years, that it would not have been so pleasurable and complete without your presence.

"I am my scars! My destiny is my own! There can be no chosen one. Only we can save ourselves."

(Illidan Stormrage)

Resumo

Redes Neurais baseadas em Grafos (GNNs) são modelos recentes criados para o aprendizado de representações de nós (e de grafos), que alcançaram resultados promissores na detecção de padrões que ocorrem em dados de larga escala que relacionam diferentes entidades. Dentre esses padrões, fraudes financeiras se destacam por sua relevância socioeconômica e por apresentarem desafios particulares, tais como o desbalanceamento extremo entre as classes positivas (fraudes) e negativas (transações legítimas), e o desvio de conceito (i.e., propriedades estatísticas dos dados mudam ao longo do tempo). Como as GNNs são baseadas em propagação de mensagem, a representação de um nó acaba sendo muito impactada pelos seus vizinhos e pelos hubs da rede, amplificando os efeitos do desbalanceamento. Pesquisas recentes tentam adaptar estratégias de subamostragem e sobreamostragem para GNNs a fim de mitigar esse efeito sem, contudo, considerar o desvio de conceito. Neste trabalho, realizamos uma série de experimentos para avaliar técnicas existentes de detecção de fraudes em rede, considerando os dois desafios anteriores. Para isso, utilizamos conjuntos de dados reais, complementados por dados sintéticos criados a partir de uma nova metodologia introduzida aqui. Também propomos um novo framework de modelo denominado GMU-GNN, que realiza a sobre-amostragem dos nós do grafo pertencentes à classe minoritária de forma a melhorar a representatividade e expressividade no espaço latente de características interpretado pelo modelo de classificação de nós. Em novos experimentos realizados com 5 datasets, o GMU-GNN obteve um desempenho superior aos demais modelos tidos atualmente como estado-da-arte sob esses mesmos contextos e propósitos do problema aqui abordado.

Palavras-Chave: Detecção de Fraude, Comportamento Fraudulento, Redes Neurais baseadas em Grafos.

Abstract

Graph-based Neural Networks (GNNs) are recent models created for learning representations of nodes (and graphs), which have achieved promising results when detecting patterns that occur in large-scale data relating different entities. Among these patterns, financial fraud stands out for its socioeconomic relevance and for presenting particular challenges, such as the extreme imbalance between the positive (fraud) and negative (legitimate transactions) classes, and the concept drift (i.e., statistical properties of the data change over time). Since GNNs are based on message propagation, the representation of a node is strongly impacted by its neighbors and by the network's hubs, amplifying the imbalance effects. Recent works attempt to adapt undersampling and oversampling strategies for GNNs in order to mitigate this effect without, however, accounting for concept drift. In this work, we conduct experiments to evaluate existing network fraud detection techniques, considering the two previous challenges. For this, we use real datasets, complemented by synthetic data created from a new methodology introduced here. We also propose a new model framework called GMU-GNN, which performs the oversampling of graph nodes belonging to the minority class in order to improve the representativeness and expressiveness in the latent space of features interpreted by the node classification model. In new experiments carried out with 5 datasets, the GMU-GNN obtained a performance superior to the other models currently considered as state-of-the-art under the same contexts and purposes of the problem addressed here.

Keywords: Fraud Detection, Fraudulent Behavior, Graph Neural Networks.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

The volume of financial transactions has surged in recent years with the rise of cryptocurrencies, digital banking and online payment gateways. These technologies have transferred control from banks to users, making the purchase and transfer process more distributed and accessible to the general population, but also more susceptible to fraud. This issue affects the performance metrics and financial results of many companies, as the total value of fraud can sometimes exceed their revenue or make the profit margin so small (even if the value of fraud is small compared to legitimate transactions) and, in the long run, make bankruptcy just a matter of time.

There are many known ways to commit fraud for which preventive actions are already taken, but at the same time, criminals are developing other new ways to circumvent these security systems. In this work, we will be addressing two distinct ways to commit fraud: review fraud and financial fraud. For review scams, some examples come in the form of sponsored comment spam, to get a boost in the ratings of a trade, or even sabotage, to get a downgrade in the ratings of an opposing trade. For financial fraud, some examples come in the form of credit card theft, terrorist financing, tax evasion, and money laundering.

To solve this problem, companies have been investing heavily in fraud detection and prevention through a set of strategies and processes to ensure the veracity and security of the financial transactions that involve them. According to Global Market Insights[1], in 2018 this market had an associated value of over 20 billion dollars and the trend is still growing. Following this logic, it is expected that in 2025 this market will have an associated value volume of more than 80 billion dollars, both in investment and in potential recovery or savings for the company, demonstrating the financial importance of this area for socioeconomic metrics.

Many research efforts have been done in this area, but most of it focuses on detecting fraudulent credit card transactions. The use of widely known supervised learning models,

[1] `gminsights.com/industry-analysis/fraud-detection-and-prevention-market`

such as Decision Trees and Support Vector Machine (SVM) (Şahin and Duman, 2011), Artificial Neural Networks and Probabilistic Graphical Models (Maes et al., 2002), and even data mining techniques (Brause et al., 1999; Chawla, 2010), has been a common thread in this field of research.

Fraud detection has proven to be a much more complex domain than one might first imagine as pointed by Abdallah et al. (2016). The design of anti-fraud systems has the additional challenges, compared to traditional forecasting systems, of limiting the number of false alarms (i.e., avoiding a high rate of false positives) and dealing with the decrease in the accuracy of predictions as your training data become obsolete due to concept drift. In data mining and machine learning, concept drift refers to the phenomenon of having the underlying data distribution change over time, which makes it very difficult to maintain the quality of predictions. Abbass et al. (2004) try to mitigate this concept drift problem using online classification techniques by adapting the loss function to be more aware of the data distribution difference between recent and older data points.

Another challenge is the unbalanced nature of the datasets in this problem, as there is far less fraudulent than legitimate instances, which makes it even more difficult for supervised learning techniques to infer the intrinsic distribution of the data. This makes it even harder for current supervised learning techniques to learn its intrinsic data distribution, making researchers to focus on gathering more fraudulent data (Maes et al., 2002).

Recently, an important relationship between entities involved in financial transactions that goes beyond first-order "source-destination" relationships has been incorporated into graph learning techniques to detect strong relationships between distinct data instances through edge weights using graphs (Weber et al., 2018; Wagner, 2019; Weber et al., 2019; Hooi et al., 2016), but none of them attempt to mitigate dataset imbalance or concept drift issues. These problems are addressed in other research works, but without trying to detect these relationships using structured graph models (Wang et al., 2017, 2021; Yang and King, 2009; Gao et al., 2020; Shamshirband et al., 2014). Initial steps have been taken recently to solve the concept drift problem using convolutional graph networks in Pareja et al. (2020) without, however, considering the problem of unbalanced classes in their evaluations. On the other hand, other recent works (Liu et al., 2021; Zhao et al., 2021) try to mitigate the data imbalance problem also using graph-based neural networks, but without taking into account the concept drift.

The main motivation for using graph-based neural networks to detect fraud is precisely the plurality of relationships found in the transaction network and which are usually great indicators for an entity belonging to a class or not. For example, these relationships can take the form of clusters, where several entities belonging to the same class also have several links to each other, causing unclassified entities that are included in this cluster to have a high probability of belonging to the same class as the others. These relationships are essential for the detection of fraud in some more specific cases in which we have intermediaries for transactions between two or more fraudulent entities, since initially they are not directly connected.

The difficulty of developing a research with usage of graph learning in financial transaction data and comparing it to other ones lies in the scarce availability of public datasets, due to this kind of data having its intrinsic need of sensitive information, making it almost impossible to publish without violating the General Data Protection Regulation (GDPR) and its deviations and adoptions around the world. In that way, recent studies have tried to replicate the distributions in each dimension of real-world data but also embedding it into synthetic dataset by using simulations Lopez-Rojas et al. (2016).

1.1 Objectives

Given the importance of detecting fraudulent transactions among legitimate ones, we intend to accomplish the objectives below:

- Model transactions by using a new graph learning technique that mitigates the imbalanced dataset and concept drift problems;
- Address the need of public datasets for future research by building and making public datasets that are used in this work;
- Develop a Graph Neural Network (GNN) model architecture capable of extracting information from tabular and structural features to accurately predict both fraudulent and legitimate classes;
- Evaluate existing techniques for detecting financial fraud in light of the two intrinsic challenge: concept drift and data imbalance.

In summary, the main objective of this dissertation is to investigate the overall performance of different state-of-the-art models in various contexts by using real and synthetic datasets, as well as proposing a new model architecture to deal with the downsides of those models and possibly achieving better results.

1.2 Contributions

The difficulty of conducting a research on graph learning for financial transaction data and comparing it with previous works lies in the scarce availability of public datasets that can be modified into a network. More precisely, this dissertation brings the following contributions:

- Proposes a new methodology to assess the robustness of financial fraud detection methods based on synthetic data generation, for a holistic view of their strengths and weaknesses;
- Quantifies the advances achieved by recent research in the field by implementing them and comparing them with previously developed models;
- Finds and proposes points of improvement for the models considered to be state-of-the-art;
- Proposes a new model (GMU-GNN) to detect fraud in a graph structured way;
- Compares the proposed model with the other existing baselines.

From our experiments it was possible to conclude that the literature in the field is indeed making strides towards better performance. However, the efforts that consider the combination of the two biggest challenges in fraud detection from transactional data are still lacking, as no model has been developed at the time of this writing.

1.3 Dissertation outline

This dissertation is organized in such way that its chapters can be read independently. This section is intended to contextualize them and help the reader decide the best way to read this document.

Chapter 2 describes the main research works done in machine learning for fraud detection, both using traditional models and graph neural network architectures.

In order to provide a basic understanding of graph neural networks, Chapter 3 presents some technical background used in this work. It also describes in details the latest state-of-the-art models used as baselines when comparing our proposed model architecture.

Chapter 4 describes the datasets used in details, including (i) some data exploration, (ii) comparison to show that each dataset captures particular characteristics. In addition, it presents a brief description for each model used in our experiments.

Chapter 5 presents the proposed model and its innovative aspects, including the key research question of this work: is possible to solve both the data imbalance and concept drift problems in a fraud detection context? It also describes each component of our new architecture in detail.

Chapter 6 describes how the experimental settings were done and the comparison results with the other baselines. We analyze those results, highlighting general trends of each model for each context, as well as their main differences.

Finally, Chapter 7 concludes this dissertation, revisiting our objectives and contributions and presents future work for further research in fraud detection.

Chapter 2

Related Work

In this chapter we present some of the most relevant work related to the detection of fraudulent behavior and categorize it according to the type of approach: traditional machine learning and graph-based learning. At the end of the chapter, we discuss our work in the context of the state of the art.

2.1 Fraud Detection through Traditional Machine Learning

One of the first models for automatic detection of fraudulent transactions was proposed more than two decades ago (Gopinathan et al., 1998). That patent describes a system proposed to extract discriminative characteristics about illicit transactions based on learned relationships between pre-selected variables known to be informative in the fraud detection domain. These relationships allow the system to estimate a fraud probability for each transaction through a neural network. The approach also produces explanations for a given prediction, since it can also issue certain codes expressing reasons that reveal the relative contribution of each characteristic to a given result, and be robust to concept drift, since its performance is monitored and the model is retrained when it falls below a predetermined threshold.

Many other techniques based on feature engineering have been proposed more recently, some of them being Dal Pozzolo et al. (2017); Carcillo et al. (2017); Lebichot et al. (2020); Carcillo et al. (2019); Le Borgne and Bontempi (2021). However the most common fraud topic investigated by these is the credit card theft, in which case the data is not usually modeled as a graph with some kind of relations between the dataset rows used to train and test these models. In that way, this contextual simplification is usually something that we do not see in real-life applications, making it difficult to find a practical way to apply those models. This is due to fraudulent behavior may involve various types of entities (e.g., individuals, merchants, products, services) and transactions (e.g., purchase, delivery, transfer, cash in, cash out). Thus, in some domains, can only be identified by considering the structure that connects a set of those entities through multiple transactions.

2.2 Fraud Detection via Graph Learning

Weber et al. (2018) is one of the few works that addresses fraud detection using graph learning techniques. It is based on the construction of Graph Convolutional Networks, a recent type of deep neural networks, introduced by Kipf and Welling (2017); Atwood and Towsley (2016) and further enhanced in their scalability by Chen et al. (2018). In Weber et al. (2018), the authors propose two distinct architectures to deal with the money laundering problem using a GCN (Graph Convolutional Network) and a FastGCN (Fast Graph Convolutional Networks) to be trained on a synthetic dataset (AMLSim). This dataset simulates a series of legitimate banking transactions mixed with a set of suspicious money laundering patterns, causing some synthetically generated features to behave like outliers, with the total transaction value being much higher than the average. They also concluded that the among the two architectures, FastGCN is best suited for this particular dataset in that it can be trained in less time for a fixed amount of epochs.

After this initial study, the authors of this research decided to address Bitcoin money laundering scenarios using graph learning techniques (Weber et al., 2019), such as Skip-GCN, a Convolutional Graph Network containing jump connections via sharing edge weights between distinct neural network depth levels, and EvolveGCN, by Pareja et al. (2020), a graph-based convolutional network architecture that handles the common temporal changes of these types of data sets (concept drift) using techniques of long-term memory. They concluded in Weber et al. (2019) that fraud detection is a difficult problem due to its intrinsic complex characteristics, given that a traditional Random Forest model was able to overcome all proposed graph-based learning methods by just providing it with the same input data, raising the possibility of a future improvement of the model. From this observation, a set of GCNs was created to mimic the behavior of a random forest. However, the authors also showed that these new models (Skip-GCN and EvolveGCN) are able to achieve better performance than its predecessor, the GCN, as shown in Table 2.1. The top part of the table shows results for methods that do not leverage the graph structure for different sets of features: LF refers to the local features, i.e., each node only knows its direct neighbors features, AF refers to all features, i.e., each node knows all other nodes features, and NE refers to the node embeddings computed by GCN.

Method	Illicit Transactions Precision	Recall	F_1	MicroAVG F_1
Logistic Regression$^{\text{AF}}$	0.404	0.593	0.481	0.931
Logistic Regression$^{\text{AF+NE}}$	0.537	0.528	0.533	0.945
Logistic Regression$^{\text{LF}}$	0.348	0.668	0.457	0.920
Logistic Regression$^{\text{LF+NE}}$	0.518	0.571	0.543	0.945
Random Forest$^{\text{AF}}$	0.956	0.670	0.788	0.977
Random Forest$^{\text{AF+NE}}$	0.971	0.675	0.796	0.978
Random Forest$^{\text{LF}}$	0.803	0.611	0.694	0.966
Random Forest$^{\text{LF+NE}}$	0.878	0.668	0.759	0.973
Multi-Layer Perceptron$^{\text{AF}}$	0.694	0.617	0.653	0.962
Multi-Layer Perceptron$^{\text{AF+NE}}$	0.780	0.617	0.689	0.967
Multi-Layer Perceptron$^{\text{LF}}$	0.637	0.662	0.649	0.958
Multi-Layer Perceptron$^{\text{LF+NE}}$	0.682	0.578	0.626	0.986
GCN	0.812	0.512	0.628	0.961
Skip-GCN	0.812	0.623	0.705	0.966

Table 2.1: Illicit transactions classification results.

	GCN			EvolveGCN		
	Precision	Recall	F1	Precision	Recall	F1
Illicit Transactions	0.812	0.623	0.705	0.850	0.624	0.720
MicroAVG	0.966	0.966	0.966	0.968	0.968	0.968

Table 2.2: GCN versus EvolveGCN performances.

Also, Table 2.2 shows the performance comparison between GCN and EvolveGCN. These methods are evaluated with respect to illicit transactions metrics (with the fraud label as the only positive label) and micro average (MicroAVG) metrics, that calculate metrics for both datasets by counting the total true positives, false negatives and false positives across all labels.

A more recent study by Liu et al. (2021) has made a significant advance in being able to adapt and utilize graph-based neural networks for highly unbalanced datasets applied in the context of network data fraud detection. In this study, a sampling step is performed at the graph vertices in order to balance the number of vertices belonging to each class and facilitate the propagation of the signal from the minority class to the nearest vertices. Using this strategy, the authors concluded that this strategy effectively improves the performance of the models when applied to this problem context.

Also, Zhao et al. (2021) tried to solve this data imbalance problem by implementing a well-known up-sampling technique called SMOTE (Synthethic Minority Over-sampling Technique) published by Chawla et al. (2002), but in a graph neural network embedding setting. This is much more complicated because we cannot guarantee that those embedded features are on a convex space, i.e., for any two data points in a subset, we cannot ascertain that this subset contains the whole line segment that joins them. This strips SMOTE interpolation calculations of theoretical backing as it can generate features that are not even close to the data samples we pass to it, inserting some unnecessary noise into the data that can cause considerable performance drops. Aware of this problem, the authors extended this over-sampling algorithm for graph neural networks domain and implement an end-to-end model that is better than other traditional models.

2.3 Relationship of this work with the State of the Art

This dissertation compares the performance of traditional approaches with that of graph-based approaches in the financial fraud detection, taking into account class imbalance and concept drift, which are characteristics inherent in this problem. This work also focus on proposing a new graph neural network model architecture to deal with those problems and evaluate them against some already existing state-of-the-art models.

To the best of our knowledge, graph learning techniques that try to tackle both imbalanced and concept-drifted datasets problems have not yet been proposed in the literature. Therefore, the success of the project could greatly advance model architecture in the area, which currently performs below expectations, as shown in Liu et al. (2021) and Zhao et al. (2021).

Chapter 3

Technical Background

As this dissertation builds on a body of works in a lot of previous knowledge about deep learning, we dedicate this chapter to present a basic introduction on this topic. We also present the theory underlying Graph Neural Networks (GNNs) and Generative Adversarial Networks (GANs) for a better understanding of each component of our proposed model.

3.1 Perceptron

The perceptron model was develop by Rosenblatt (1958) inspired by previous researches about logical calculus in human nervous activity (McCulloch and Pitts, 1943). Although we currently use much more complex artificial neural networks, the Perceptron provides us a clear understanding of how a neural network works in mathematical terms.

The perceptron takes as inputs n real numbers $x_1, x_2, \ldots, x_n$ and outputs a binary value. The output is calculated by applying a non-linear function to a weighted sum of the inputs parameterized by weights $w_1, w_2. \ldots, w_n$, as shown in Figure 3.1, i.e.,

$$\text{output} = \begin{cases} 0, \text{ if } \phi(\sum\limits_{0}^{n} w_n x_n) \leq \text{ threshold} \\ 1, \text{ if } \phi(\sum\limits_{0}^{n} w_n x_n) > \text{ threshold} \end{cases} \tag{3.1}$$

where ϕ can be any non-linear function (e.g., Heaviside, Sigmoid, or Rectified Linear Unit – ReLU).

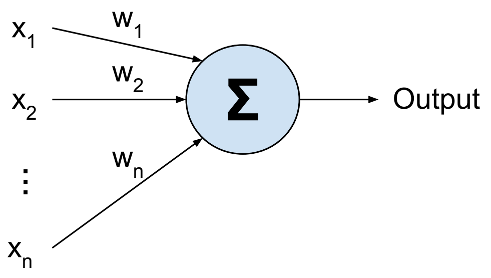


Figure 3.1: Schema of a Perceptron

3.2 Multi-layer Perceptron (MLP)

The basic Perceptron structure was not able to solve more complex problems, such as the famous XOR operator. Later on, Rumelhart et al. (1986) found out that a more complex structure that interconnects multiple units and layers could solve most of the open complex problems by learning an approximate function to map inputs to the output. A few years later, Cybenko (1989) proved that any function could be sufficiently approximated by a Multi-layer Perceptron with a Sigmoid activation function. This is known as the Universal Approximation Theorem. Later on, Hornik (1991) proved that it is not the specific choice of an activation function, but instead the multilayer feed-forward architecture itself that is responsible for giving neural networks the potential of being universal approximators.

A convenient way to express the calculations of a Multi-layer Perceptron is by using the vector notation. The perceptron computation can be expressed as

$$\text{output} = \begin{cases} 0 & \text{if } \phi(\vec{w}^\top \vec{x} + \vec{b}) \leq 0 \\ 1 & \text{if } \phi(\vec{w}^\top \vec{x} + \vec{b}) > 0 \end{cases} \tag{3.2}$$

with ϕ being any non-linear function (activation function), $\vec{w}^\top$ as the transposed weights vector, $\vec{x}$ as the features vector, and $\vec{b}$ being the biases vector that can be also expressed as the first weight of each neuron, i.e., w_0 associated with the input 1. Also, a detailed schema can be seen in Figure 3.2 for easier visualization.

This classic Perceptron layer structure is still used is recent state-of-the-art models, but is usually described as a Fully Connected Layer.

3.3 Graph Convolutional Networks

Kipf and Welling (2017) introduced a similar approach from the framework of graph convolutions known from spectral graph theory (Bruna et al., 2014; Henaff et al., 2015) to define

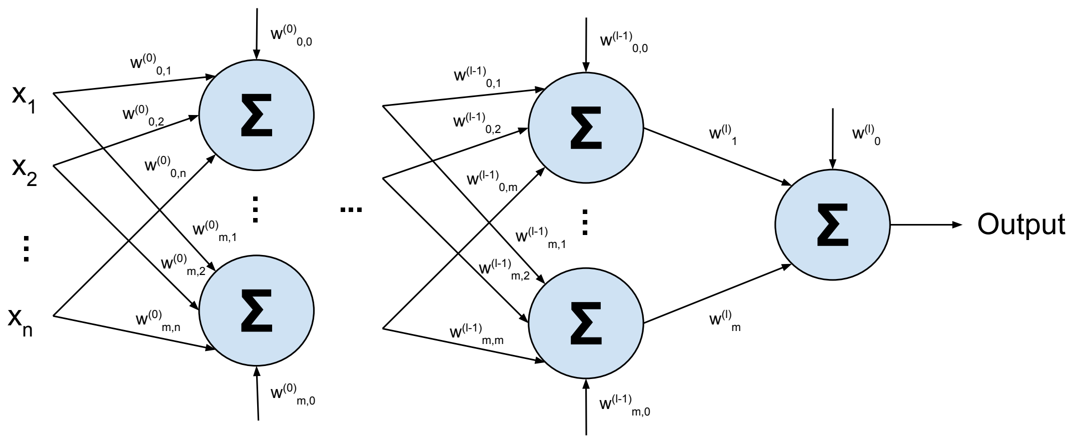


Figure 3.2: Schema of a Multi-layer Perceptron

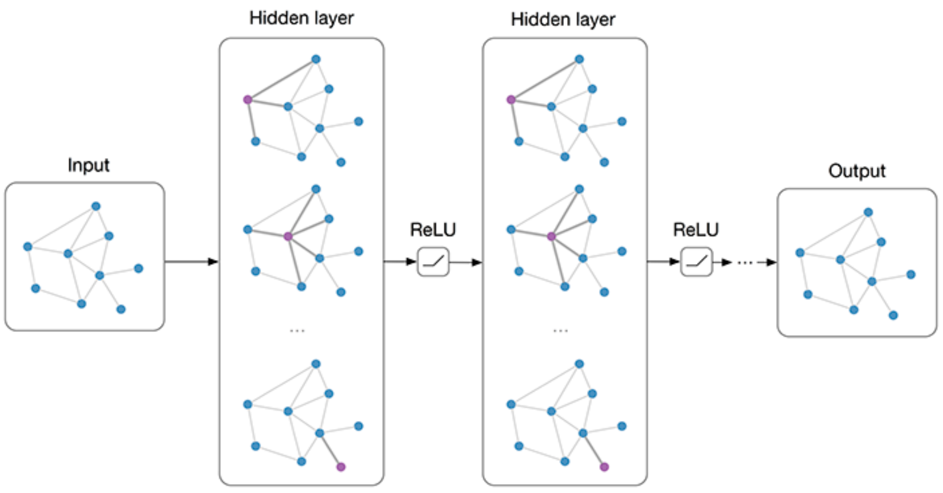


Figure 3.3: Schema of a Graph Convolutional Network

parameterized filters used in a multi-layer neural network model and from the approximations of smooth filters in the spectral domain using Chebyshev polynomials with free parameters that are learned in a neural network-like model architecture (Defferrard et al., 2016). However, Kipf and Welling (2017) introduce simplifications that in many cases allowed both faster training times and higher predictive accuracy than the previous methods on a given set of graph datasets benchmark. A schema example of this model can be seen in the Figure 3.3.

This new model was called Graph Convolutional Networks (GCNs), with the term "convolutional" being due to the fact that filter parameters are typically shared over all locations in the graph or in one of its subgraphs, as in Duvenaud et al. (2015). For these models, the main objective is to learn a function that represents features on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which takes as input a feature description x_n for every node n, expressed in a NxD feature matrix X, with N being the number of nodes and D the number of input features, and a representative description of the graph structure in matrix form, usually an adjacency matrix A or some derivation of it. These inputs produces a node-level output Z.

Similarly, as presented in 3.1 and 3.2, every neural network layer can be expressed as a non-linear function

$$H^{(l)} = f(H^{(l-1)}, A), \tag{3.3}$$

with $H^{(0)}$ being the input features, $H^{(L)} = Z$, and L as the number of layers.

One of the simplest forms of a layer-wise propagation rule in an GCN is expressed as

$$H^{(l+1)} = f(H^{(l)}, A) = \sigma(AH^{(l)}W^{(l)}), \tag{3.4}$$

with $W^{(l)}$ being the weight matrix for the l-th neural network layer and $\sigma(\cdot)$ the non-linear activation function (e.g., ReLU).

Also, Kipf and Welling (2017) address two major limitations of this simple model. The first one is that the multiplication with A means that, for every node, it sums up all the feature vectors of all neighbors nodes but not the node itself. They fixed this problem easily by adding the identity matrix to A, enforcing self-loops for each node. The second one is that A is typically not normalized, resulting in a complete change of scale of the feature vector, potentially inserting arithmetical overflows or underflows. They also fixed this by normalizing A such that all rows sum to one, i.e., $D^{-1}A$, where D is the diagonal node degree matrix, being equivalent to taking the average of neighboring node features. In practice, they used a more complex symmetric normalization $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$.

Thus, the resulting propagation formula of a GCN can be expressed as

$$f(H^{(l)}, A) = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right), \tag{3.5}$$

with $\hat{A} = A + I$, where I is the identity matrix and $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$.

3.4 Generative Adversarial Networks

Fraud datasets are usually very unbalanced when comparing the number of legitimate with illicit data volume. In that sense, we need a way to create some balance between those classes, even if artificially. Goodfellow et al. (2014) proposed a solution to create fake data that resembles the real data by training a Generative Adversarial Network (GAN). This model architecture trains a neural network known as generator to create fake data using random noise inputs by competing against another neural network known as discriminator, as illus-

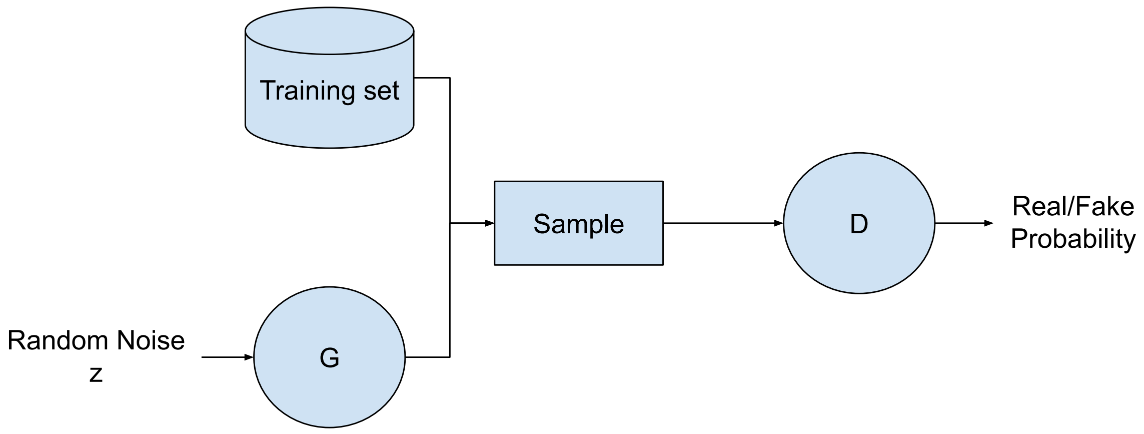


Figure 3.4: Schema of a Generative Adversarial Network training step

trated in Figure 3.4. These two agents play a zero-sum game, i.e., the discriminator gain is the generator loss and vice-versa. This game can be expressed as

$$\min_{\gamma} \max_{\delta} \mathbb{E}[\log(D_\delta(x)) + \log(1 - D_\delta(G_\gamma(z)))], \tag{3.6}$$

with δ being the discriminator D parameters and γ being the generator G parameters.

However this vanilla model architecture does not handled tabular data well. Later, Xu et al. (2019) proposed a new model that was capable of generating synthetic tabular data and Wang et al. (2018) proposed a new model that was capable of generating synthetic graphs.

3.5 Problem Formulation

In this section, we formalize the graph-based fraud detection problem, also presenting the definitions and notations used.

Definition 1 (Multi-Relational Graph) *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X}, \mathcal{C})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of vertices (nodes); $\mathcal{E} = \{\mathcal{E}_1, \ldots, \mathcal{E}_R\}$ is the set of edges, separated by R relations; $\mathcal{X}$ and $\mathcal{C}$ are, respectively, the feature sets and vertex labels, so that for each node $v_i \in \mathcal{V}$, $x_i \in \mathcal{X}$ is a d-dimensional feature vector and $c_i \in \mathcal{C}$ is its binary label, $\forall\, i = \{1, \ldots, N\}$.*

Definition 2 (Imbalance Ratio) *Given a set of labels $\mathcal{C}$ where $c_i \in \{1, 2\}$, we can define a partition $(\mathcal{C}_1, \mathcal{C}_2)$ of $\mathcal{V}$ such that $C_k = \{v_i \in \mathcal{V} : v_i = k\}$. The Imbalance Ratio between $\mathcal{C}_1$ and $\mathcal{C}_2$ is defined as $IR = \frac{|\mathcal{C}_1|}{|\mathcal{C}_2|} \in [0, +\infty)$. If $IR > 1$, $\mathcal{C}_1$ is called the majority class and $\mathcal{C}_2$ the minority class. When $IR = 1$, it is said that $\mathcal{C}$ is perfectly balanced.*

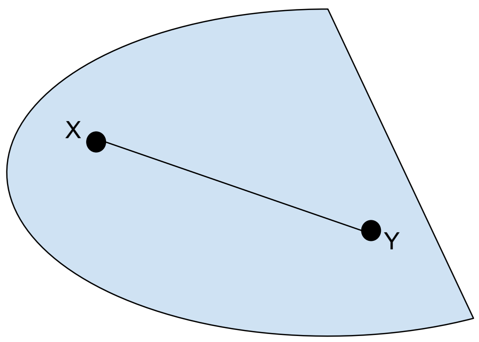


Figure 3.5: Illustration of a convex region

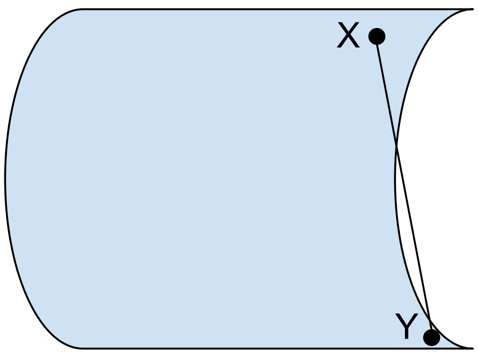


Figure 3.6: Illustration of a non-convex region

Definition 3 (Concept Drift) *In data analysis and machine learning, a concept drift occurs when the statistical properties associated with the target variable c (label) change over time. This causes problems because the predictions of a given model become less accurate over time. Concept drift is quite common in real world data, considering that many of them are directly linked to mappings of human behavior and nature, which, in turn, are not governed by any deterministic mathematical law, causing gradual changes in their distribution.*

Definition 4 (Graph-based Fraud Detection) *The graph-based fraud detection problem is defined on the multi-relational graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X}, \mathcal{C})$, from the Definition 1. Each v_i vertex has been statically labeled c_i (fraudulent or legitimate) associated with its x_i characteristics. Thus, this problem consists of predicting the label of unlabeled vertices contained in the graph based on their transactional features, in order to find similarities and disparities between characteristics of vertices belonging to the same class and distinct classes, respectively.*

For a better understanding of the data interpolation problem to generate new data point candidates based on raw and embedded features, we will need those next two definitions:

Definition 5 (Convex Region) *In geometry, a subset of an Euclidean space is **convex** if, for any two points in the subset, the whole line segment that joins them is contained in the same subset. Equivalently, a **convex set** or **convex region** is a subset that intersects every line into a single line segment. An example of a convex region can be seen in Figure 3.5.*

Definition 6 (Non-Convex Region) *Similarly, a subset of an Euclidean space is **non-convex** or **concave** if there exists any two points in the subset for which the whole line segment that joins them is not contained in the same subset. Equivalently, a **non-convex***

set *or* ***non-convex region*** *is a subset that does not intersects every line into a single line segment. An example of a non-convex region can be seen in Figure 3.6.*

Chapter 4

Methodology

In this section we present the datasets and models used in this project.

4.1 Datasets

For this study, we used some publicly available datasets, each one with a property that will help us evaluate the model performance under a given context.

4.1.1 Synthetic data generated with Anti-money Laundering Simulator (AMLSim)

The AMLSim[1] (Pareja et al., 2020; Weber et al., 2018) is a multi-agent simulator proposed to generate dataset for the fraud detection domain based on another more generic transaction simulator (it does not have any specificity to generate fraud behavior), called PaySim (Lopez-Rojas et al., 2016). From AMLSim, three datasets were generated, as shown in Table 4.1, in order to study the robustness of existing methods in relation to class imbalance and concept deviation. These datasets are generated based on different parameterization files, which are used to generate data that mimics various legitimate and fraudulent transactions.

The first dataset, AMLSim 1, is perfectly balanced and will serve as a baseline for the others. The second, AMLSim 2, has a strong unbalance of classes and will serve to demonstrate the impact of this phenomenon on the performance of the studied models. The third, AMLSim 3, was also generated with a high rate of unbalance between classes, but with an additional concept drift effect, defined with more details in Definition 4.1.3. More precisely, the average values of legitimate and illicit transactions are varied month-to-month,

[1] `https://github.com/IBM/AMLSim/`

Dataset	Total Transactions	Legit Transactions	Illicit Transactions	Transactions IR	Total Entities	Legit Entities	Illicit Entities	Entities IR	Concept Drift?
AMLSim 1	10,000	5,000	5000	1	1,000	500	500	1	No
AMLSim 2	10,000	9,500	500	19	1,000	950	50	19	No
AMLSim 3	10,000	9,500	500	19	1,000	950	50	19	Yes
Elliptic Data Set	234,355	49,215	4,687	10.5	203,769	42,019	4,545	9.25	Yes
YelpCHI	67,395	58,479	8,916	6.56	38,063	30,325	7,738	3.92	No

Table 4.1: AMLSim, Elliptic Data Set and YelpCHI datasets statistics.

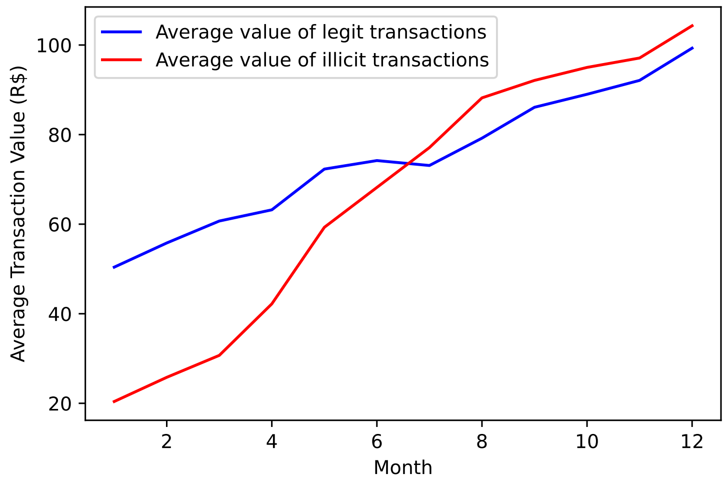


Figure 4.1: Variation of the average cash value of legitimate and illicit transactions over a year for AMLSim 3.

during 12 months of simulation, as can be seen in Figure 4.1. Also note that the average values even cross after month 7.

4.1.2 Elliptic Data Set

This dataset was collected for about two weeks directly from the Bitcoin cryptocurrency blockchain and maps transactions from real anonymized entities to legitimate ones (stock exchanges, portfolio providers, miners, licit services, etc) or as illicit (scams, malwares, terrorist organizations, ramsomwares, etc) (Weber et al., 2019). It contains 203,769 vertices (transaction agents) and 234,355 edges (financial transactions). Among the vertices, only 23% of these vertices are labeled, being 2% illicit (4,545 nodes) and 21% (42,019 nodes) legitimate, while the remaining 77% have no labeling at all, as shown in Table 4.1.

There are 166 characteristics associated with each node of the graph, however, due to privacy concerns, an exact description of all of them was not provided. The observation time was discretized into 49 evenly spaced windows. Each node is associated with a single time window and its edges represent each transfer, deposit or withdrawal operation, representing a measure of the time a transaction was transmitted to the Bitcoin network. Each time window contains a single connected component of transactions that appeared in the blockchain in less than three hours of each other and there are no edges connecting the different time steps.

The first 94 characteristics represent local information about the transaction, including the time step described above, number of entries and exits, transaction rate, exit volume and other aggregate numbers such as average Bitcoins received and spent. The remaining 72 characteristics are aggregations obtained using transaction information from hops between the central node, giving the maximum, minimum, standard deviation and correlation coefficients of neighboring transactions for the same information data (number of inputs/outputs, transaction rate, etc).

4.1.3 YelpCHI

This dataset was collected from the yelp.com website that consists of 67,395 reviews for a set of 201 hotels and restaurants in the Chicago area and contains information about the products and their 38,063 users, timestamps , ratings, and free text review (Mukherjee et al., 2013; Rayana and Akoglu, 2015, 2016), as shown in Table 4.1. Yelp has a built-in filtering algorithm that identifies suspicious reviews and stores them in a separate list of legitimate reviews. Although this filter is not perfect, it has produced accurate results in previous studies (Weise, 2011). Therefore, your results can be used as labels for certain reviews.

In this dataset, we have 58,479 (86.77%) legitimate ratings, 8,916 (13.23%) filtered ratings, 30,325 (79.67%) legitimate users, and 7,738 (20.33%) suspicious users. Thus, the imbalance ratio of user labels is 3.92, indicating a moderate imbalance when compared to the other datasets used in this research. To assemble the graph, only the User-Review-Product (U-R-P) relationships were used, so that each user is linked to their reviews of each product by means of edges.

4.2 Baseline Models

In the experiments, we use some implementations of classical models, which are not able to incorporate information about the relationship between nodes:

- XGBoost (Chen and Guestrin, 2016);
- CatBoost (Prokhorenkova et al., 2018).

in addition to some graph-based neural network models as baselines:

- GCN (Kipf and Welling, 2017): uses first-order local approximation of the convolution of spectral graphs;
- GAT (Veličković et al., 2018): uses attention mechanism for neighbor aggregation;
- GraphSAGE (Hamilton et al., 2017): uses induction on a fixed-size sample of neighboring nodes;
- GraphSAINT (Zeng et al., 2020): using graph node sampling for greater scalability and efficiency.

and, finally, some models of neural networks based on graphs as state of the art:

- EvolveGCN (Pareja et al., 2020): uses a combination of graph convolutions with long-term memory techniques to deal with gradual temporal changes in data (concept drift);
- PC-GNN (Liu et al., 2021): uses a new sampling method to propagate the signal from the minority class to neighboring nodes. This is done in three steps: sampling, choosing and aggregating. The central nodes are sampled with a class balancer in order to build a balanced sub-graph for the training stage. Under a parameterized distance measure, the neighborhood of each node belonging to the minority class is oversampled, while the neighborhood of each node belonging to the majority class is undersampled. Then, an aggregation of message propagations from different sub-graphs is made to obtain a final representation of the complete graph;
- GraphSMOTE (Zhao et al., 2021): uses a well known over-sampling technique (SMOTE) that interpolates embedding features from the minority class to generate newer nodes in order to balance the data and then uses a vanilla GCN or GraphSAGE model to perform the node classification task.

For this dissertation, I implemented, in an end-to-end manner, both GCN and PC-GNN models. I also adapted XGBoost, CatBoost, GAT, GraphSAGE, GraphSAINT, EvolveGCN, and GraphSmote models to work on our datasets. All the code used in this project is publicly available in a repository[2].

[2]`https://github.com/ronaldpereira/brasnam-experiments`

Chapter 5

Proposed Model: GMU-GNN

In this chapter we describe GMU-GNN (**G**enerative **M**inority **U**p-sampling **G**raph **N**eural **N**etwork), our proposed neural network architecture for graph-based fraud detection. We start by doing a deep-dive into the many details of the proposed architecture, including each structure that is contained in the model framework. Then, we discuss the novelty of our approach with respect to already existing ones in the literature.

5.1 Model Architecture

Many studies which try to solve some variant of the fraud detection problem face the challenges of having a highly unbalanced and concept drift-prone dataset. In that way, the vast majority of it work by up-sampling of the minority class and adapting the model loss function as an attempt to address these challenges altogether. In this dissertation, we follow these guidelines, but using different approaches.

GMU-GNN uses three different stacked models, each one of them having a specific objective to solve. The first one, CTGAN (Xu et al., 2019), is used for up-sampling the minority class raw features and generating new nodes. The second one, GraphGAN (Wang et al., 2018), is used for creating a new graph that contains those new nodes. The third one, GCN (Kipf and Welling, 2017) or GraphSAGE (Hamilton et al., 2017), is used for accomplishing the node classification task.

5.1.1 CTGAN – Conditional Tabular Generative Adversarial Network

To solve the class imbalance problem in a node level, we need to generate new nodes that contain features associated to the minority class. This could seem simple at a first glance, but as pointed out by Chawla et al. (2002) and Xu et al. (2019), generating values in a non-convex feature space, as defined in Definition 6, is very challenging and the interpolation can generate some anomalies along the way. Thus, using the non-convex embedded feature space for any up-sampling is not advised as it can generate some feature values that are too far away from the original distribution.

Thereby, we use a different approach to this problem. Specifically, we perform the up-sampling by generating new nodes (features and labels) in the raw feature space, that is contained in a convex region, as defined in Definition 5. In that way, we use CTGAN to generate those new nodes by using the minority class features.

CTGAN uses a mode-specific normalization technique to deal with columns with complex distributions. According to Xu et al. (2019), for each continuous-valued column C_i, it uses a Variational Gaussian Mixture Model (VGM) (Bishop, 2006) to estimate the number of modes m_i and fit a Gaussian mixture. An example of this calculation is a three-modal VGM ($m_i = 3$), denoted by η_1, η_2 and η_3, as the learned Gaussian mixture can be expressed as

$$\mathbb{P}_{C_i}(c_{i,j}) = \sum_{k=1}^{3} \mu_k \, \mathcal{N}(c_{i,j}; \eta_k, \phi_k), \tag{5.1}$$

where μ_k and ϕ_k are the weight and standard deviation of a mode m_k, respectively. After this step, for each row $c_{i,j}$ in C_i, compute the probability of $c_{i,j}$ coming from each mode. For instance, the probability densities are ρ_1, ρ_2 and ρ_3 and computed as

$$\rho_k = \mu_k \, \mathcal{N}(c_{i,j}; \eta_k, \phi_k), \tag{5.2}$$

where ρ_k is the probability density of a mode m_k. Finally, it samples one mode according to the probability densities and uses the sampled mode to normalize the value. For example, we pick the third mode given ρ_1, ρ_2 and ρ_3. Then we can represent $c_{i,j}$ as a one-hot vector $\beta_{i,j} = [0, 0, 1]$ indicating the third mode and a scalar $\alpha_{i,j} = \frac{c_{i,j} - \eta_3}{4\phi_3}$ to represent the value within the mode. Then, the representation of a row becomes the concatenation of continuous and discrete columns

$$\mathbf{r}_j = \alpha_{1,j} \oplus \beta_{1,j} \oplus \cdots \oplus \alpha_{N_c,j} \oplus \beta_{N_c,j} \oplus \mathbf{d}_{1,j} \oplus \cdots \oplus \mathbf{d}_{N_d,j}, \tag{5.3}$$

where $\mathbf{d}_{i,j}$ is the one-hot representation of a discrete value.

In short, CTGAN is a generative adversarial network that receives a tabular input for its training step and can generate newer samples via a random noise input when requested in a prediction step. This is beneficial because, for every dataset that we use in this project, every table row denotes a different node in the graph. Thus, for every N new samples generated

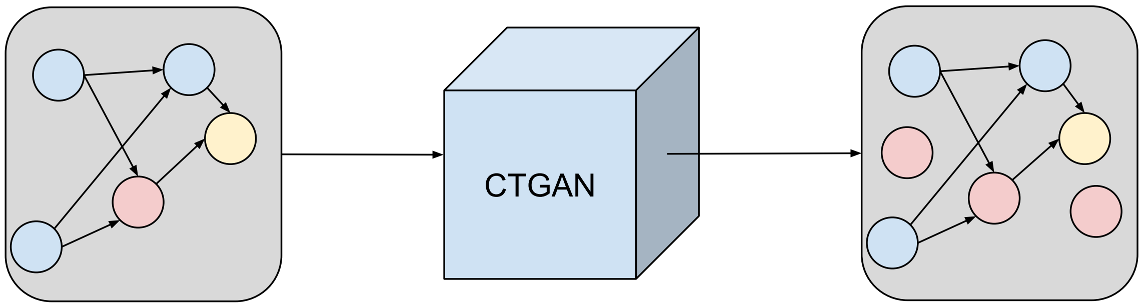


Figure 5.1: Illustration of the use case of a Conditional Tabular Generative Adversarial Network (CTGAN)

by CTGAN, we have N new nodes that we can use to augment the graph, as shown in Figure 5.1.

5.1.2 GraphGAN – Graph Generative Adversarial Network

In this step, we face the the node linking problem, in which nodes generated in previous steps are not linked to any already existing node in the graph, because CTGAN only treats generation in the raw feature space. To solve it, we then use GraphGAN (Wang et al., 2018) to generate and create new graphs that contains edges that links those new nodes to pre-existing ones without losing the basic node linking structure with respect to the original graph.

The intuition behind GraphGAN is quite similar to the presented in Section 3.4. The only difference is that the training set data contains all the nodes from the original graph and the basic graph structure via edges representation (e.g., $[1, 2]$ indicates that nodes 1 and 2 are linked). Then, the discriminative model objective is to determine which graphs are real and fakes, thus influencing the generative model to improve its graph generation capabilities each training epoch. This results in a graph that contains both generated and original nodes linked by generated and original edges, as it can be seen in Figure 5.2.

It is important to mention that GraphGAN only creates new edges if one of the nodes are new nodes generated by CTGAN in the previous step. In that way, the basic structure of the original graph is maintained as-is as a way to not change the information passed through its edges.

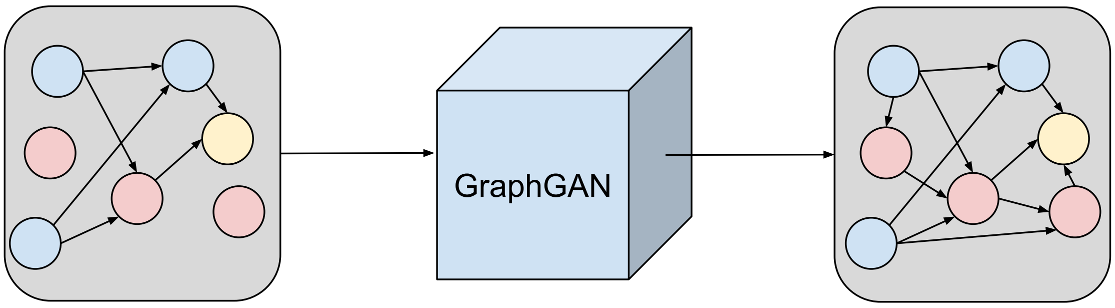


Figure 5.2: Illustration of the use case of a Graph Generative Adversarial Network (GraphGAN)

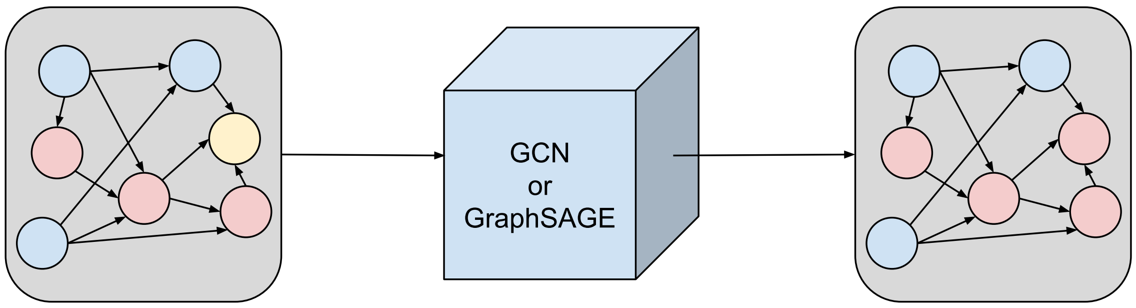


Figure 5.3: Illustration of the use case of a node classification model (GCN or GraphSAGE)

5.1.3 GCN – Graph Convolutional Network

GCN (Kipf and Welling, 2017) is used as a node classification model exactly as already defined and presented in Section 3.3 and can be seen in Figure 5.3.

5.1.4 GraphSAGE – Graph Sample and Aggregate

GraphSAGE (Graph SAmple and aggreGatE) was first introduced in Hamilton et al. (2017) as a model architecture to solve the inherent graph convolutional network transductive setting problem. It assumes that the entire graph is available at training time, but without the labels of the validation and test sets nodes. This naturally causes a higher difficulty for the model to generalize unseen nodes, which are a common case in some real-world applications.

In that context, they introduce this model architecture as an inductive setting framework that leverages node feature information to efficiently generate node embeddings for previously unseen data. This is done by learning a function that generates embeddings via

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

```
Input  : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth $K$; weight matrices
         $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity $\sigma$; differentiable aggregator functions
         $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$
Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$
1 $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;
2 for $k = 1...K$ do
3     for $v \in \mathcal{V}$ do
4         $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
5         $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
6     end
7     $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
8 end
9 $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$
```

Figure 5.4: GraphSAGE embedding generation algorithm

sampling and aggregating features from a node local neighborhood instead of training individual embeddings for each node.

As presented in Hamilton et al. (2017) in Figure 5.4: “The intuition behind Algorithm 1 is that at each iteration, or search depth, nodes aggregate information from their local neighbors, and as this process iterates, nodes incrementally gain more and more information from further reaches of the graph”. In this dissertation, we used the element-wise max-pooling aggregator function, which is both symmetric and trainable. In this approach, each neighbor’s vector is independently fed through a fully-connected neural network such as

$$\text{AGGREGATE}_k^{pool} = \max(\{\sigma(\mathbf{W}_{pool}\mathbf{h}_{u_i}^k + \mathbf{b}), \forall\, u_i \in \mathcal{N}(v)\}), \tag{5.4}$$

where $\max$ denotes the element-size max operator, σ is a non-linear activation function, $\mathbf{W}_{pool}$ is the pooled weight matrix, $\mathbf{h}_{u_i}^k$ is the hidden layer features for each layer k, u_i is an uniform draw $\mathcal{N}(v)$ from the set $\{u \in \mathcal{V} : (u, v) \in \mathcal{E}\}$, and $\mathbf{b}$ is the bias vector, and . Also, an illustration of the usage of this model can be seen in Figure 5.3.

5.1.5 GMU-GNN – Generative Minority Up-sampling Graph Neural Network

After presenting all the blocks that compose the GMU-GNN architecture above, we can proceed to explain how it all fits together into a single model. At the training step, the training set tabular data is given as an input to the model framework. It then calculates the Imbalance Ratio (IR), as defined in Definition 2, to find which class is the minority one.

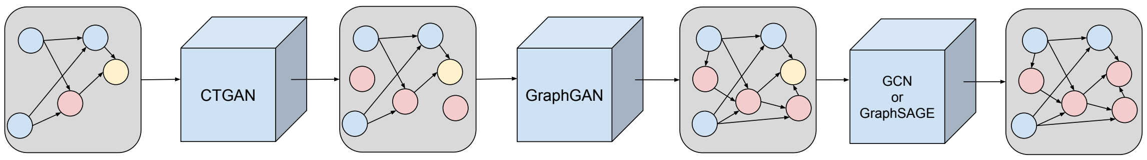


Figure 5.5: Illustration of GMU-GNN training step

After this step, all the labeled training data that belongs to the minority class is then given to the CTGAN model (Xu et al., 2019), that, in turn, generates N additional synthethic rows, which correspond to N synthethic nodes not yet linked to the graph. To connect the to other nodes, we use GraphGAN (Wang et al., 2018) to generate and create new graph edges that links those nodes without losing the basic structural information of the original graph. Then, the final step is to use this minority up-sampled graph to train a node classification model. We consider either GCN (Kipf and Welling, 2017) or GraphSAGE (Hamilton et al., 2017), to output the probability of the node belonging to each of the classes. An illustration of this step can be seen in Figure 5.5.

5.2 Differences to previous approaches

GMU-GNN is a stack of three distinct models, which we combine to tackle the problems of data imbalance and concept drift. The first part is based on CTGAN (Conditional Tabular Generative Adversarial Networks) from Xu et al. (2019) whose objective is to synthetically generate new data entries for the minority class, creating both features and labels associated with it. The second part is based on GraphGAN (Graph Generative Adversarial Networks) from Wang et al. (2018) aiming to integrate the synthetic data generated in the previous step with the rest of the input graph, generating edges that carry some information with it. Finally, the last part is based on GCN (Graph Convolutional Networks) from Kipf and Welling (2017) and GraphSAGE from Hamilton et al. (2017) in order to perform the node classification task for fraud detection.

To the best of my knowledge, many other models that address class imbalance problem either use a statistical-based oversampling modeling (Zhao et al., 2021; Liu et al., 2021; Zeng et al., 2020) or a copy-based oversampling task, i.e., generating repetitions of the exact same minority class data points by a multiplier N before inputting the data into the model training step (Carcillo et al., 2019; Lebichot et al., 2020; Chawla, 2010), but only a few of them attempt to generate newer data with GANs (Lei et al., 2020; Zola et al., 2020). Therefore, by performing this synthetic generation with generative models, we were able to perform a broader search through the latent feature space by exploring new data points not previously contained in the original dataset.

In the graph embedding space, features are in a non-convex region, as defined in Definition 6, making it very hard to interpolate between two given data points and generate new synthetic samples (Zhao et al., 2021). Thus, using a raw feature space to generate this samples is simpler and can be more reliable, as they have a higher probability of belonging to a convex space. This is the main reason why we used two complementary steps to generate synthetic data points for our dataset: one for the raw tabular feature space (CTGAN) and one for the graph structure embedding space (GraphGAN).

Chapter 6

Experimental Settings and Results

In this chapter we describe the experiments conducted in this project to evaluate all baseline models and our proposed model, as well as the discussions for each experiment result. We begin by explaining each one of the experiments done in this dissertation. Then, we expose the experimental setup, including how the data was split into training, validation and test sets and how the hyperparameters were chosen via tuning. Next, we test the reliability of AMLSim synthethic dataset generation by comparing it versus a real world dataset. Finally, we compare and discuss the results achieved in the experiments between the baselines and our proposed model.

6.1 Experiments

The first experiment conducted in this dissertation was a proof that AMLSim (Weber et al., 2018, 2019) is reliable enough to be a valid dataset considered in this project benchmark. For that purpose, we did a real world dataset generation test by comparing it with the public dataset YelpChi (Mukherjee et al., 2013; Rayana and Akoglu, 2015, 2016). For this test, we gathered the star rating feature, the user node degree and labels distribution from YelpChi dataset and passed as hyperparameters for AMLSim tool to check if the data distribution for the generated dataset in similar to the real world one.

The second experiment conducted in this dissertation was to assess the robustness of the models in relation to class imbalance. For that purpose, we performed experiments using the two synthetic datasets, AMLSim 1 and AMLSim 2, whose TD imbalance rates are equal to 1 (perfectly balanced) and 19 (extremely unbalanced), respectively.

The third experiment was to evaluate the robustness of the methods to a gradual class concept drift, we performed experiments using the synthetic dataset AMLSim 3. We used months 1 to 4 to define the training dataset, while the remaining data was used to define two test sets: the first containing months 5 to 8, and the second containing months 9 to 12. As explained in Definition 3, this data suffers from a change in the distribution of its characteristics month by month, so that the greater the distance between the training months and the testing months, the lower the expected performance of the model.

Finally, the fourth experiment was to test the real world context performance for those models. To achieve this, we used two real world public datasets: YelpChi (Mukherjee et al., 2013; Rayana and Akoglu, 2015, 2016) and Elliptic Data Set (Weber et al., 2019).

6.2 Experimental setup

To measure and compare the performance of the models, we use two metrics widely adopted in classification problems, F1-macro and AUC, in addition to a metric specifically proposed for fraud detection problems, called F1-fraud.

The F1-macro is the simple arithmetic mean of the F1-score of each class. F1-fraud, on the other hand, is the F1-score considering only instances labeled as fraud as a positive class. Although correlated with the first, this metric emphasizes the precision and recall dimensions specifically for the class of our greatest interest. The third metric is the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve and indicates the probability that a random positive instance of the test set is labeled as more likely to be fraudulent than a random negative instance of the same set. An important feature of the AUC is that it is independent of the probability threshold chosen to classify instances as negative or positive, thus even if the other classification metrics were relatively lower and the AUC is higher, we can easily do a threshold adaptation to make the model performance better.

6.2.1 Train-validation-test split

To train and evaluate each model in this project, we did a train-validation-test split by getting 80% of the data for the training set, 10% for the validation set, and 10% for the test set. The training set is the dataset that goes for the model learning in the train step, thus we cannot expect it to be evaluated as the model performance. For this exact purpose in this step, we separated a validation set that serves to be a model performance evaluation during the training. This was also used to verify convergence in models as we can calculate the loss for each epoch and stop the training process earlier than specified if it hasn't improved for more than N epochs. Finally, when the training step is ended, we have a data the model has never seen, known as the test set, that we calculate metrics for it, thus becoming our true model performance evaluation result.

Hyperparameter	Search Space	Best Hyperparameters Found for Dataset				
		YelpChi	Elliptic Data Set	AMLSim 1	AMLSim 2	AMLSim 3
CTGAN hyperparameters						
Epochs	[10, 500]	98	187	267	289	447
Number of Samples	{0, 1000, 2000, 3000, 5000}	3000	1000	0	3000	5000
Generator Learning Rate	$\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$	10^{-3}	10^{-4}	10^{-2}	10^{-2}	10^{-5}
GraphGAN hyperparameters						
Epochs	[10, 500]	277	254	189	341	486
Generator Learning Rate	$\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$	10^{-4}	10^{-2}	10^{-1}	10^{-2}	10^{-3}
GCN hyperparameters						
Learning Rate	$\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$	10^{-2}	10^{-1}	10^{-1}	10^{-3}	10^{-3}
Epochs	[10, 1000]	207	187	193	419	818
Number of Units in Hidden Layer 1	{12, 16, 20, 24, 30, 50}	20	30	30	16	30
Dropout	{0.1, 0.2, 0.3, 0.4, 0.6, 0.8}	0.2	0.3	0.4	0.3	0.1
GraphSAGE hyperparameters						
Learning Rate	$\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$	10^{-3}	10^{-3}	10^{-2}	10^{-2}	10^{-4}
Epochs	[10, 1000]	61	122	144	288	457
Dropout	{0.1, 0.2, 0.3, 0.4, 0.6, 0.8}	0.2	0.4	0.3	0.1	0.2

Table 6.1: GMU-GNN hyperparameter search space and best values found for each dataset

6.2.2 Hyperparameter Tuning

In order to set GMU-GNN's hyperparameters, we applied a Randomized Search with Cross Validation (RandomizedSearchCV) from Pedregosa et al. (2011), as it was already proved to be more efficient that grid-based searches in Bergstra and Bengio (2012). The search space and best parameters chosen are contained in Table 6.1 and those who are not being shown are used as the model original author predefined default values.

6.2.3 Computational resources

All experiments were run on an Ubuntu 20.04 LTS whose specs were Intel(R) Core(TM) i7-4790 CPU@3.60GHz, 4 cores, 2 threads per core, Nvidia(R) GTX 1660 6 GB GPU, and 16 GB RAM.

6.3 Results

In this section we will describe the results of the experiments made in this research. We start by testing the reliability of AMLSim synthetic dataset generation by comparing it against a real world dataset distribution and then we present the comparison between baselines and our GMU-GNN proposed model.

6.3.1 AMLSim Synthetic Generation Reliability Test

For our first experiment, we proved that AMLSim (Weber et al., 2018, 2019) is reliable enough to be a valid dataset considered in this project benchmark by comparing it with the public dataset YelpChi (Mukherjee et al., 2013; Rayana and Akoglu, 2015, 2016). In that way, by gathering the distribution of star rating feature, the user node degree and labels distribution from YelpChi dataset and passing it as hyperparameters for AMLSim tool, we could see that the density distribution for each of those variables were pretty similar, as shown in Figure 6.1. Also, we did some sample graph generation for random five users in YelpChi and AMLSim datasets, and they were also very similar, as the graphs shown in Figure 6.2.

6.3.2 Comparison between Baselines and Proposed Model

To compare our proposed model against all the baselines presented in Section 4.2, we did some experiments by using all datasets described in Section 4.1.

Performance Degradation due to Class Imbalance: Table 6.2 presents the performance of each model for the second experiment conducted in this dissertation to address performance degradation due to class imbalance. Analyzing only AMLSim 1, it is observed that state-of-the-art models for fraud detection can be surpassed by other methods when the data is balanced. For our proposed model this is not true, given that by not detecting a class imbalance, minority class up-sampling does not happen. Thus, the performance of GMU-GNN on balanced datasets is very close to its base node classification models (i.e., $\text{GMU-GNN}_{\text{GCN}} \approx$ GCN and $\text{GMU-GNN}_{\text{GraphSage}} \approx$ GraphSAGE) and can be seen as an ablation test.

On the other hand, the results in AMLSim 2 show that when there is an imbalance in the data, the performance of the models degrades a lot. Even so, the GMU-GNN performed significantly better than the other models, demonstrating the ability to adapt to the current context.

Degradation over time due to Concept Drift: Table 6.3 shows the results obtained for the two test sets in order to evaluate the robustness of the models to a gradual class concept drift. Analyzing the results for the first test set (AMLSim months 5 to 9), it is observed that

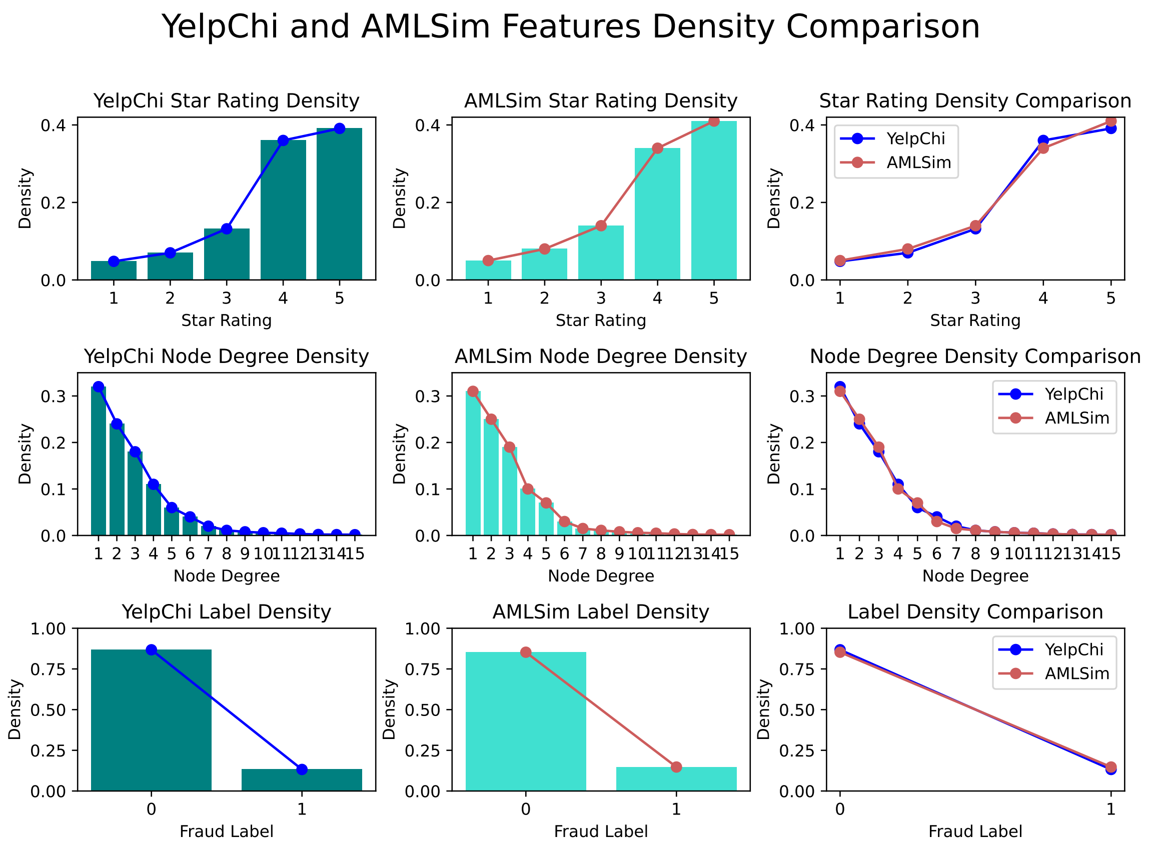


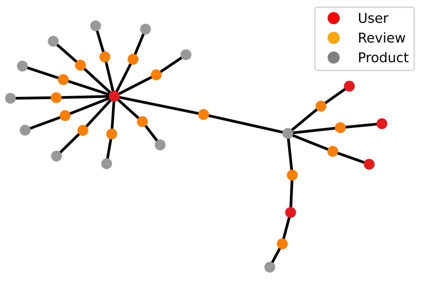
Figure 6.1: Comparison between YelpChi and AMLSim datasets

the models had a low overall performance in terms of F1-macro and F1-fraud, but the AUC obtained by some models remained close to or above 0.7. The explanation for this is that, although the classification threshold defined based on the training set makes many errors in the test cases, the ranking of nodes according to the probability of being fraudulent remains reasonably good, causing the AUC not to degrade this much. In this case, a fraud detection system could be "fixed" with a simple adjustment of this threshold.

On the other hand, the results for the second test set (AMLSim months 10 to 12) show a different result. Even with the drop in F1-macro, F1-fraud and AUC of all models tested here, the GMU-GNN still manages to perform better than the others, attesting to the model's resilience in relation to the concept drift problem.

Also, there is a curious factor even to be analyzed for the traditional and basic models of Graph Neural Networks: all of them were below 0.5, indicating that such methods usually classify negative instances as more likely to be frauds than positive instances. This is explained by the crossing of the curves in Figure 4.1, which makes these models consider instances as belonging to the opposite class.

Real-World Application: For our last result, Table 6.4 shows the performance of each



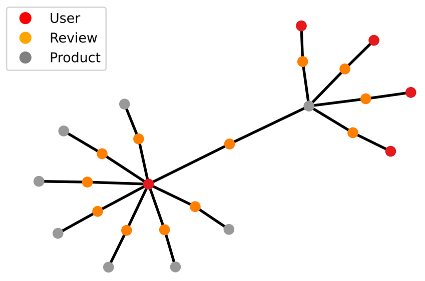


Figure 6.2: YelpChi 5 users sample graph and AMLSim 5 users generated graph

Dataset	AMLSim 1 (IR = 1)			AMLSim 2 (IR = 19)		
Metric	F1-macro	F1-fraud	AUC	F1-macro	F1-fraud	AUC
XGBoost	.908 ± .015	.866 ± .002	.863 ± .015	.551 ± .019	.114 ± .010	.529 ± .015
CatBoost	**.934** ± **.019**	.879 ± .012	.909 ± .018	.628 ± .015	.172 ± .013	.578 ± .019
GCN	.779 ± .001	.853 ± .016	.927 ± .006	.548 ± .015	.206 ± .006	.550 ± .012
GAT	.840 ± .011	.846 ± .017	.879 ± .019	.549 ± .008	.164 ± .010	.597 ± .010
GraphSAGE	.933 ± .014	.880 ± .008	**.933** ± **.016**	.556 ± .009	.181 ± .006	.546 ± .018
GraphSAINT	.878 ± .011	.821 ± .017	.909 ± .009	.561 ± .005	.235 ± .016	.559 ± .007
EvolveGCN	.753 ± .019	.645 ± .006	.906 ± .013	.713 ± .006	.307 ± .011	.811 ± .015
PC-GNN	.739 ± .013	.662 ± .018	.879 ± .009	.722 ± .019	.413 ± .007	.832 ± .013
GraphSMOTE	.900 ± .018	.832 ± .016	.931 ± .017	.743 ± .009	.421 ± .017	.831 ± .010
GMU-GNN$_{\text{GCN}}$	.781 ± .006	.851 ± .027	.924 ± 0.05	.782 ± .017	.470 ± .015	.858 ± .018
GMU-GNN$_{\text{GraphSAGE}}$	.931 ± .013	**.881** ± **.011**	**.933** ± **.012**	**.783** ± **.017**	**.478** ± **.011**	**.864** ± **.008**

Table 6.2: Baseline and proposed models performance comparison in synthetic datasets AMLSim 1 and AMLSim 2

model in the two real data sets (YelpChi and Elliptic Data Set) as a function of the three evaluation metrics previously presented. The most recent methods, considered state-of-the-art – EvolveGCN, PC-GNN and GraphSmote –, proved to be more effective than the traditional ones. However, even so the GMU-GNN managed to be comparatively superior in its performance, attesting to the applicability in the real world of the model proposed in this project.

Dataset	AMLSim 3 (IR = 19) Months 5, 6, 7 and 8			AMLSim 3 (IR = 19) Months 9, 10, 11 and 12		
Metric	F1-macro	F1-fraud	AUC	F1-macro	F1-fraud	AUC
XGBoost	.513 ± .017	.104 ± .016	.542 ± .010	.290 ± .010	.196 ± .013	.311 ± .017
CatBoost	.417 ± .014	.179 ± .005	.519 ± .017	.376 ± .018	.236 ± .012	.369 ± .009
GCN	.479 ± .012	.229 ± .019	.567 ± .006	.254 ± .010	.000 ± .000	.366 ± .007
GAT	.530 ± .007	.226 ± .011	.645 ± .018	.177 ± .008	.000 ± .000	.315 ± .020
GraphSAGE	.463 ± .017	.161 ± .005	.636 ± .009	.366 ± .006	.091 ± .013	.432 ± .005
GraphSAINT	.464 ± .008	.148 ± .011	.659 ± .015	.357 ± .016	.119 ± .005	.454 ± .014
EvolveGCN	.613 ± .011	.416 ± .014	.723 ± .015	.444 ± .007	.312 ± .015	.601 ± .018
PC-GNN	**.699 ± .017**	**.517 ± .015**	.733 ± .011	.507 ± .007	.308 ± .010	.560 ± .015
GraphSMOTE	.602 ± .020	.377 ± .009	.745 ± .011	.547 ± .019	.318 ± .010	.580 ± .008
GMU-GNN$_{\text{GCN}}$	.597 ± .006	.482 ± .017	.765 ± .015	**.558 ± .009**	.354 ± .018	.716 ± .010
GMU-GNN$_{\text{GraphSAGE}}$	.607 ± .016	.480 ± .014	**.776 ± .019**	.556 ± .016	**.367 ± .006**	**.719 ± .014**

Table 6.3: Baseline and proposed models performance comparison in synthetic concept drift dataset AMLSim 3

Dataset	YelpChi			Elliptic Data Set		
Metric	F1-macro	F1-fraud	AUC	F1-macro	F1-fraud	AUC
XGBoost	.522 ± .010	.267 ± .019	.647 ± .015	.561 ± .006	.238 ± .013	.578 ± .015
CatBoost	.559 ± .018	.216 ± .018	.612 ± .020	.548 ± .014	206 ± .013	.560 ± .015
GCN	.560 ± .005	.246 ± .008	.613 ± .015	.496 ± .015	.069 ± .008	.565 ± .008
GAT	.556 ± .006	.128 ± .007	.614 ± .010	.467 ± .006	.056 ± .018	.562 ± .015
GraphSAGE	.515 ± .018	.257 ± .007	.675 ± .010	.479 ± .008	.074 ± .012	.570 ± .006
GraphSAINT	.534 ± .005	.270 ± .019	.673 ± .014	.505 ± .011	.359 ± .017	.629 ± .015
EvolveGCN	.583 ± .005	.429 ± .011	.740 ± .013	**.680 ± .014**	.265 ± .007	.739 ± .011
PC-GNN	.629 ± .011	.386 ± .011	.747 ± .006	.661 ± .014	.234 ± .008	.677 ± .006
GraphSMOTE	.669 ± .007	.376 ± .019	.745 ± .008	.652 ± .007	.253 ± .015	747 ± .010
GMU-GNN$_{\text{GCN}}$	.645 ± .015	**.449 ± .007**	**.759 ± .013**	.672 ± .006	.342 ± .018	.782 ± .019
GMU-GNN$_{\text{GraphSAGE}}$	**.695 ± .014**	.431 ± .019	.758 ± .006	.677 ± .005	**.372 ± .011**	**.804 ± .018**

Table 6.4: Baseline and proposed models performance comparison in real world datasets YelpChi and Elliptic Data Set

Chapter 7

Conclusions and Future Works

In this chapter we summarize this dissertation's contributions to the state-of-the-art knowledge on fraud detection using Graph Neural Networks. We also present some points that might be further investigated in future works.

7.1 Conclusion

This dissertation is an effort towards generating solutions to the barely explored landscape of detecting fraudulent behavior on transactions by using structural data. In that way our primary objective was to develop a Graph Neural Network (GNN) model architecture capable of extracting information from tabular and structural features regarding transactional data to accurately predict both fraudulent and legitimate classes. In order to benchmark it against other baselines, we evaluate them on the same datasets, as a way to identify which one performs better at each use case.

At the same time, we address the need for public datasets for future research by building and making publicly available many synthetic datasets that are used in this work to train and evaluate our proposed models under different scenarios, by simulating unbalanced data and the concept drift phenomenon. Moreover, we showed that AMLSim is a reliable tool to generate those datasets. Specifically, we generated a synthethic dataset using the same distributions as the YelpChi dataset and verified that the results were pretty similar for some of the most important characteristics as they have almost the same density distribution in both.

Also, the models proposed by Liu et al. (2021) and Zhao et al. (2021) achieved good predictive performance on different sets of datasets. But, ideally, these models should be general enough to yield good results for a broad range of datasets. In this context, the present work also aims to evaluate existing techniques for detecting financial fraud in light of the two intrinsic challenges (concept drift and data imbalance) using publicly available datasets, two real datasets – YelpChi and Elliptic Data Set – and three synthetically generated through a transaction simulator called AMLSim (Pareja et al., 2020; Weber et al., 2018). These results were published in Pereira and Murai (2021).

Finally, we propose a new graph learning architecture called Generative Minority Upsampling Graph Neural Network (GMU-GNN) for detecting fraudulent transactions based on two Generative Adversarial Networks: CTGAN (Xu et al., 2019) to generate synthetic nodes and GraphGAN (Wang et al., 2018) to create a new graph that includes edges linking those new nodes. The proposed technique mitigates the imbalanced dataset and is robust to concept drift. Also, we showed that this model outperforms both existing traditional and state-of-the-art models presented in Section 4.2 on real world and synthetic datasets. Another important result achieved is that our model also performs well on different contexts: for a balanced dataset, it performed as well as the baseline models – GCN (Kipf and Welling, 2017) and GraphSAGE (Hamilton et al., 2017) –, in contrast to previous state-of-the-art models, that performed worse than traditional models.

7.2 Future Works

In this dissertation we foresee that three possible avenues that could be pursued to improve the results presented here. The first one relates to investigating how to solve the significant drop in performance caused by datasets that exhibit concept drift characteristics. Many preliminary experiments has been done by changing the loss function associated with the node classification models to a mean aggregator function in GraphSAGE, as the behavior is similar to skip connections, which can lead to a better generalization, as pointed out by Hamilton et al. (2017). However, this made the oversmoothing problem (i.e., node labels converge to a single class in the entire graph) to happen much more quicker than it usually happened by using a max aggregator function.

The second one is related to automatically selecting the best candidates generated by both CTGAN (Xu et al., 2019) and GraphGAN (Wang et al., 2018), as they still generates some anomalies (e.g., nodes that does not resemble any existing class and graphs that does not resemble the original graph structure) during the training process. A possible direction to solve this problem may be appending the discriminator model to each generator to rate the samples generated by them, to compute a type of reliability score. Then, all samples below a predefined threshold will be disregarded in the training step and will be discarded, avoiding those obvious anomalies that can be easily detected by a pretrained discriminator.

The final one is contextualized in a more recent problem, that is detecting fraudulent patterns that takes place across different institutions without exchanging any sensible information about specific individuals, as each institution only has access to data associated with its customers. This is an important research topic as new data regulation laws – GDPR (General Data Privacy Regulation) and LGPD (Lei Geral de Proteção de Dados Pessoais) – are enforcing this.

Bibliography

Abbass, H. A., Bacardit, J., Butz, M. V., and Llora, X. (2004). Online adaptation in learning classifier systems: stream data mining. *IlliGAL report*, 2004031.

Abdallah, A., Maarof, M. A., and Zainal, A. (2016). Fraud detection system: A survey. *J NETW COMPUT APP*, 68:90–113.

Atwood, J. and Towsley, D. (2016). Diffusion-convolutional neural networks. In *NeurIPS*, pages 1993–2001.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag New York.

Brause, R., Langsdorf, T., and Hepp, M. (1999). Neural data mining for credit card fraud detection. In *ICTAI*, pages 103–106. IEEE.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2014). Spectral networks and locally connected networks on graphs. In *ICLR*.

Carcillo, F., Dal Pozzolo, A., Le Borgne, Y.-A., Caelen, O., Mazzer, Y., and Bontempi, G. (2017). Scarff : a scalable framework for streaming credit card fraud detection with spark. *Information Fusion*, 41.

Carcillo, F., Le Borgne, Y.-A., Caelen, O., Kessaci, Y., Oblé, F., and Bontempi, G. (2019). Combining unsupervised and supervised learning in credit card fraud detection. *Information Sciences*, 557:317–331.

Chawla, N. V. (2010). *Data Mining for Imbalanced Datasets: An Overview*, pages 875–886. Springer US, Boston, MA.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *JAIR*, 16:321–357.

Chen, J., Ma, T., and Xiao, C. (2018). Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *ICLR*.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *KDD*, pages 785–794.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., and Bontempi, G. (2017). Credit card fraud detection: A realistic modeling and a novel learning strategy. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–14.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *NIPS*, 29:3844–3852.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. In *NeurIPS*, volume 28.

Gao, L., Yang, L., Arefan, D., and Wu, S. (2020). One-class classification for highly imbalanced medical image data. In *Medical Imaging 2020: Imaging Informatics for Healthcare, Research, and Applications*, volume 11318, page 113181C. International Society for Optics and Photonics.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. *NeurIPS*, 63(11):139–144.

Gopinathan, K. M., Biafore, L. S., Ferguson, W. M., Lazarus, M. A., Pathria, A. K., and Jost, A. (1998). Fraud detection using predictive modeling. US Patent 5,819,226.

Hamilton, W. L., Ying, R., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *NeurIPS*, pages 1025–1035.

Henaff, M., Bruna, J., and LeCun, Y. (2015). Deep convolutional networks on graph-structured data. In *ICLR*.

Hooi, B., Song, H. A., Beutel, A., Shah, N., Shin, K., and Faloutsos, C. (2016). Fraudar: Bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, page 895–904.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257.

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

Le Borgne, Y.-A. and Bontempi, G. (2021). *Machine Learning for Credit Card Fraud Detection - Practical Handbook*. Machine Learning Group (Université Libre de Bruxelles - ULB).

Lebichot, B., Le Borgne, Y.-A., He-Guelton, L., Oblé, F., and Bontempi, G. (2020). Deep-learning domain adaptation techniques for credit cards fraud detection. In Oneto, L., Navarin, N., Sperduti, A., and Anguita, D., editors, *Recent Advances in Big Data and Deep Learning*, pages 78–88, Cham. Springer International Publishing.

Lei, K., Xie, Y., Zhong, S., Dai, J., Yang, M., and Shen, Y. (2020). Generative adversarial fusion network for class imbalance credit scoring. *Neural Computing and Applications*, 32(12):8451–8462.

Liu, Y., Ao, X., Qin, Z., Chi, J., Feng, J., Yang, H., and He, Q. (2021). Pick and choose: A gnn-based imbalanced learning approach for fraud detection. In *WWW*, page 3168–3177, New York, NY, USA. Association for Computing Machinery.

Lopez-Rojas, E., Elmir, A., and Axelsson, S. (2016). Paysim: A financial mobile money simulator for fraud detection. In *The 28th European Modeling and Simulation Symposium*, pages 249–255. Dime University of Genoa.

Maes, S., Tuyls, K., Vanschoenwinkel, B., and Manderick, B. (2002). Credit card fraud detection using bayesian and neural networks. In *1st International NAISO Congress on Neuro Fuzzy Technologies*, pages 261–270.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Mukherjee, A., Venkataraman, V., Liu, B., and Glance, N. (2013). What yelp fake review filter might be doing? *ICWSM*, 7(1):409–418.

Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T., and Leiserson, C. (2020). Evolvegcn: Evolving graph convolutional networks for dynamic graphs. *AAAI*, 34(04):5363–5370.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *JMLR*, 12:2825–2830.

Pereira, R. and Murai, F. (2021). Quão efetivas são redes neurais baseadas em grafos na detecção de fraude para dados em rede? In *Anais do X Brazilian Workshop on Social Network Analysis and Mining*, pages 205–210, Porto Alegre, RS, Brasil. SBC.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. (2018). Catboost: Unbiased boosting with categorical features. In *NeurIPS*, page 6639–6649.

Rayana, S. and Akoglu, L. (2015). Collective opinion spam detection: Bridging review networks and metadata. In *KDD*, pages 985–994.

Rayana, S. and Akoglu, L. (2016). Collective opinion spam detection using active inference. In *SDM*, pages 630–638. SIAM.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Şahin, Y. and Duman, E. (2011). Detecting credit card fraud by decision trees and support vector machines. In *IMECS*, volume 1, pages 442–447.

Shamshirband, S., Anuar, N. B., Laiha, M., Kiah, M., and Misra, S. (2014). Anomaly detection using fuzzy q-learning algorithm. *Acta Polytechnica Hungarica*, 11(8):5–28.

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

Wagner, D. (2019). Latent representations of transaction network graphs in continuous vector spaces as features for money laundering detection. In *SKILL*, pages 143–154.

Wang, G., Yang, J., and Li, R. (2017). Imbalanced svm-based anomaly detection algorithm for imbalanced training datasets. *Etri Journal*, 39(5):621–631.

Wang, H., Wang, J., Wang, J., Zhao, M., Zhang, W., Zhang, F., Xie, X., and Guo, M. (2018). GraphGAN: Graph representation learning with generative adversarial nets. *AAAI*, 32(1):2508–2515.

Wang, X., Jin, B., Du, Y., Cui, P., Tan, Y., and Yang, Y. (2021). One-class graph neural networks for anomaly detection in attributed networks. *Neural Computing and Applications*, pages 1–13.

Weber, M., Chen, J., Suzumura, T., Pareja, A., Ma, T., Kanezashi, H., Kaler, T., Leiserson, C. E., and Schardl, T. B. (2018). Scalable graph learning for anti-money laundering: A first look. *arXiv preprint arXiv:1812.00076*.

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

Weise, K. (2011). A lie detector test for online reviewers. *Bloomberg Business Week*.

Xu, L., Skoularidou, M., Cuesta-Infante, A., and Veeramachaneni, K. (2019). Modeling tabular data using conditional gan. In *NeurIPS*.

Yang, H. and King, I. (2009). Ensemble learning for imbalanced e-commerce transaction anomaly classification. In *ICONIP*, pages 866–874. Springer.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. (2020). GraphSAINT: Graph sampling based inductive learning method. In *ICLR*.

Zhao, T., Zhang, X., and Wang, S. (2021). Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *WSDM*, page 833–841, New York, NY, USA. Association for Computing Machinery.

Zola, F., Bruse, J. L., Barrio, X. E., Galar, M., and Urrutia, R. O. (2020). Generative adversarial networks for bitcoin data augmentation. *BRAINS (IEEE)*, pages 136–143.