Detecting Health Care Frauds In Attributed Graphs Using Explainable Methods.

Bastien Giles

N° d’ordre NNT :

THÈSE de DOCTORAT
DE L’UNIVERSITÉ JEAN MONNET SAINT-ÉTIENNE

Membre de l’Université de Lyon

Ecole Doctorale N°488
SIS - Sciences Ingénierie Santé

Spécialité de doctorat : Informatique

Soutenue publiquement le 25 novembre 2024, par :

Bastien GILES

Détection de fraude à l'assurance maladie à l'aide de modèles d'apprentissage automatique et de fouille de données explicables et interprétables.

Devant le jury composé de :

Cécile BOTHOREL, Maître de conférences HDR, IMT Atlantique	Rapporteure
Vincent LABATUT, Maître de conférences HDR, Université d'avignon	Rapporteur
Osmar ZAIANE, Professeur des universités, University of Alberta	Examinateur
Robardet CELINE, Professeur des universités, INSA Lyon	Examinatrice
Christine LARGERON LETENO, Professeur des universités, Université Jean Monnet Saint-Etienne	Directrice de thèse
Baptiste JEUDY, Maître de conférences, Laboratoire Hubert Curien	Co-encadrant de thèse
Damien Saboul, Docteur,	Invité

Contents

List of Figures

List of Tables

Introduction

In 2019, the Commission des Affaires Sociales du Sénat estimated that healthcare fraud losses amounted to around 1 billion euros during its inquiry at the request of La Cour des Comptes [20]. Fraud has also affected private insurance. Be-ys Group, a health data management and processing specialist for 20 years, provides health insurance fraud detection services to mutual insurance companies. These fraud detection solutions rely on a deep understanding of health professions and the availability of very large volumes of digital transactions.

Despite the effectiveness of existing systems, the constantly evolving tactics of fraudsters require continuous innovation. This thesis addresses this critical need by developing a novel system that not only detects new, previously unidentified fraud patterns but does so through explainable methods, thereby enhancing both the accuracy and transparency of fraud detection and providing actionable insights for domain experts.

Application Case

Use-case

This thesis will concentrate on a dataset for the optical specialty, but its findings are applicable to many other applications. Each insurance claim requires three actors: a healthcare provider, a prescriber, and a beneficiary

(the patient).

As a health data management and processing specialist, Be-ys is responsible for the automatisation of care payments for health insurance companies. This includes the detection of fraudulent claims. If we follow the example of optical care, we get the following actions:

- An insurance beneficiary needs glasses or lenses and goes to his ophthalmologist (Prescriber).
- The ophthalmologist details the correction needed for the equipment in a prescription that the beneficiary then brings to his optician (Provider).
- The optician follows the prescription and proposes various options, such as frames, to the beneficiary.
- This creates an insurance claim sent to the insurer for review.
- Be-ys receives this claim detailing the care selected and must validate it if it is legitimate or block it if it is detected as fraudulent.

In our case, only the domain expert can decide what constitutes fraud, but here are a few examples of the rules created for the expert system:

> Rule 48: Block claim for tinted glasses for a provider that presents an excess of tinted glasses compared to their peers.

> Rule 11: claim for glasses for all family members under the same contract but on different days in the same week.

R48 describes the over-occurrence of a rare phenomenon. R11 describes an uncommon behavior, as the family members under the same contract are almost always the children of the contractor. The normal behavior is to bring every household member on the same day and not one by one each day.

Current Detection System

Currently, the company uses an expert system to detect health insurance fraud. An expert system uses a knowledge base containing facts and rules. In our system, the facts are represented by the care request (claim), and the rules are patterns of fraud that non-IT fraud detection experts have identified. The second part of an expert system is an inference engine that applies these rules to the facts to make deductions. This system was selected for legal purposes, with domain experts legally validating the rules, producing an efficient and consistent solution where the same input always leads to the same decision.

Health Insurance Fraud

Although there are many research works on healthcare fraud insurance worldwide, notably through the Medicare dataset made available by the US government, these works report many types of fraud. Notable examples include "Uninsured individual using Medicare/Medicaid ID card of someone else to obtain services and items" [15, 21]. Still, due to differences in social policy between our countries, those frauds are almost non-existent in France [12]. Providers in France are responsible for more than 70% of the detected fraud. To better illustrate that difference, the main focus of the French report is audiologists, which are healthcare providers who require a prescription from a physician to provide care. At the same time, the main concern expressed in the US article [15] is a 98% increase in opioid drug diversion, which makes the works on those datasets hardly transferable due to the many differences between the two healthcare systems.

Data

Our only data source is the insurance claims filed by the providers, which contain identifiers for its actors and details on the care provided, such as

detailed price distributions and specifications on what the care entails. For R11 and R48, we need information not directly present in the care, such as other claims from the same contract for R11 and statistical information about the provider's peers for R48. This information is contained in other claims and aggregated in the database so that it can be added to the facts of the expert system. However, this is only possible since these patterns have already been identified before the task; classical tabular data would not allow our methods to identify these patterns without having already identified and added this information. To overcome this, we propose using relational data, i.e., graphs that can represent each claim and the relations induced by their actors with other claims.

New Frauds

In our introduction, we use the term "new frauds," this term does designate new types of frauds that result from the adaptations of fraudsters to new fraud detection methods, which would correspond to the definition of novelty:

> *Novelty detection is the task of classifying test data that differ in some respect from the data available during training.* [57]

The meaning of "new" in our application is simply a fraud not yet covered by a rule in our expert system. Thus, we will not be dealing with novelties but will need to detect elements not yet labeled as "frauds".

This also means that we are not interested in a system that can only re-detect already-identified frauds, complexifying both the identification and evaluation process.

Explainability

To allow the company to use the results of our models, we need to give the domain experts enough information on the elements detected so that they can derive a new rule to block this fraud. This gives us a bit more space compared to producing information that would be directly used for legal justifications, as it allows human intervention to interpret the results. This gives us a degree of freedom in what constitutes our explanation and also allows us a bit more leeway with false positives compared to direct blockage. However, since we are looking for unidentified frauds, we will not be able to use evaluation metrics that require full labeling. This explanation will be the only basis for the expertise that will constitute our evaluation process.

Additional Information

Additionally, to what has been presented before, we benefit from another system put in place to guide the domain expert. Our company is currently using a clustering algorithm on the aggregated statistical information of the provider, which yields a good identification of fraudsters; however, we currently encounter many troubles in identifying the exact fraud patterns they employ to block them accurately. We attribute this to the following hypothesis: "All fraudsters commit fraud for financial gain, leading to relatively easy identification, but they do not obtain this gain in the same way."

We plan to use this information as the label information to guide our model.

Scientific Context

Machine Learning

Machine learning involves training models to recognize patterns in data, enabling predictions or classifications. Two key approaches are supervised and

unsupervised learning, both of which are crucial in anomaly detection, especially in the context of healthcare fraud detection.

Supervised Learning

In supervised learning, models are trained on labeled data, where each example includes a known label. The model learns to predict these labels for new, unseen data. This approach works well when there is a substantial amount of labeled data, such as known cases of fraud. However, in fraud detection, labeled examples are often scarce, and new types of fraud may emerge, requiring the model to be frequently updated.

Semi-supervised Learning

There are various branches of supervised learning, such as semi-supervised learning a paradigm in which only a small portion of the output is known to the model for training purposes—and self-supervised learning. In this machine-learning process, the model trains itself to learn one part of the input from another part of the input.

Unsupervised Learning

Unsupervised learning, in contrast, deals with data that has no labels. The model identifies patterns and detects anomalies based on deviations from these patterns. This method is useful for discovering new or emerging fraud patterns, but it can be challenging to assess the model's accuracy since there are no labels against which to verify.

Label Availability

In our use case, supervised learning would require us to determine whether or not a claim is a new fraud. By definition, and as stated above, the new

fraud is unknown. As such, this creates a situation where we either have no information or simply intuitions derived from expertise or other models, leading us to deal with mislabeling, i.e., normal claims labeled as frauds and frauds labeled as normal. This limited portion of known information leaves us the choice only between unsupervised learning and semi-supervised learning.

Anomaly Detection

Anomalies are often referred to as outliers, i.e., observations or data points that deviate significantly from the expected pattern or the majority of the data within a dataset. However, the concept of what constitutes an anomaly has evolved over time. Initially, anomalies were primarily viewed as simple statistical outliers, but today, the term encompasses a broader range of phenomena, including errors, genuine rarity, uniqueness in the data, and, most critically, fraudulent activities.

In the context of healthcare fraud detection, anomalies are typically indicative of suspicious or aberrant behavior that may signal fraudulent actions. For example, a claim that significantly deviates from standard medical practices or contains unusual patterns may be flagged as an anomaly. Given that fraud is inherently rare and hidden within large volumes of legitimate transactions, anomalies in this domain are particularly challenging to detect.

Anomalies are, by their nature, rare occurrences. They represent deviations from the norm, which is defined by the majority of data points in a given dataset. This rarity introduces a significant challenge in anomaly detection, as it results in a severe class imbalance where the anomalous instances are vastly outnumbered by normal instances. This imbalance complicates the detection process, making it difficult for traditional methods to accurately identify these rare cases.

The distinctions between outliers, anomalies, and frauds are nuanced. Chapter 1 will explore these nuances, along with their implications for anomaly detection methods, in greater detail.

Graph

While graphs allow the representation of relationships by links between vertices corresponding to entities, attributed graphs provide, in addition, an attribute matrix that contains characteristics or features of the nodes. For example, in the case of a social network, the attributed graph describes the interactions between the users, but also the profile of each user (age, gender, center of interest, etc.) [35].

Contributions

This thesis presents several key contributions to the field of anomaly detection in graph-based machine learning. Each contribution is rigorously evaluated through extensive experiments and compared against state-of-the-art methods.

Anomaly Detection with Defective Labeling

We introduce *Suspicious*, a novel reconstruction-based weakly semi-supervised framework for anomaly detection in attributed graphs. This framework is designed to be robust against mislabeling in the training set, making it particularly effective in real-world scenarios where labeling errors are common. Extensive experiments demonstrate that *Suspicious* not only outperforms existing methods on classical graph anomaly benchmarks but also exhibits superior robustness in the presence of defective labeling.

A Simple Explanation of Graph Autoencoders (GAEs)

We propose an original explanation method based on reconstruction error that generates explanations in the same format as other state-of-the-art graph neural network explainers. This method simplifies the interpretation of graph autoencoders, making them more accessible for practical applications. Our

approach has been thoroughly evaluated and shown to rival, and in some cases exceed, the performance of more complex explainability techniques in extensive experiments.

Case Study: Real-World Application in Healthcare Fraud Detection

We apply the proposed methods in a real-world case study focused on healthcare fraud detection within the French healthcare system. This case study highlights the practical utility and effectiveness of our contributions in detecting both known and novel fraud patterns. The results obtained were compared with those from state-of-the-art methods, demonstrating the superiority of our approaches in realistic, complex environments.

Thesis Structure

Chapter 1 begins by reviewing the evolution of the concept of anomaly within graph-based machine learning. It covers the various methodologies employed for anomaly detection, starting with traditional unsupervised graph outlier detection, progressing to deep unsupervised anomaly detection techniques, and concluding with semi-supervised methods specifically designed for fraud detection. This chapter also introduces our work on *Suspicious*, a reconstruction-based weakly semi-supervised framework for anomaly detection in graphs, followed by a comprehensive evaluation of its effectiveness through extensive experiments. This work was published in a national conference [26] and an international conference [25]

Chapter 2 focuses on the explainability of graph neural networks (GNNs). It provides an overview of state-of-the-art explainability techniques. It introduces a novel method to generate explanations from the reconstruction error of graph autoencoders, using a format consistent with other leading GNN explainers. The chapter includes a thorough evaluation of these methods,

adapted from graph classification tasks to anomaly detection scenarios. This work is currently submitted to the IEEE Transactions on Knowledge and Data Engineering journal.

Chapter 3 applies the theoretical concepts and methods discussed in the previous chapters to a real-world case study in fraud detection within the French healthcare system. This chapter details the processes involved in creating the datasets. It presents the results of applying the proposed methods to detect both known and novel fraud patterns in a practical, real-world setting.

List of Publications

Bastien Giles, Baptiste Jeudy, Christine Largeron and Damien Saboul, "Reconstruction Errors: a Simple yet Efficient Explanation for Graph Autoencoder Anomaly Detection," Submitted in IEEE Transactions on Knowledge and Data Engineering (TKDE), 2024

Bastien Giles, Baptiste Jeudy, Christine Largeron and Damien Saboul, "Suspicious: a Resilient Semi-Supervised Framework for Graph Fraud Detection," 2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI), Atlanta, GA, USA, 2023, pp. 212-220, [25]

Bastien Giles, Baptiste Jeudy, Christine Largeron, Damien Saboul, "Un cadre semi-supervisé résilient pour la détection d'anomalie sur graphe attribué." In Extraction et Gestion des Connaissances (EGC) 2022, pp.55-66, [26]

Chapter 1

Detecting Anomalies in Attributed Networks

In this chapter, we will explore various approaches and methodologies for detecting anomalies in attributed networks. The chapter begins by providing an overview of the state-of-the-art techniques, discussing the evolution of unsupervised graph anomaly detection from early methods to more sophisticated approaches involving network representation. Following this, we delve into specific graph deep learning backbones, including graph convolutional networks (GCNs), graph attention networks (GATs), and autoencoders, each designed to handle the complexities of graph-structured data. The chapter also covers the integration of these methods with anomaly detection tasks, presenting both unsupervised and semi-supervised strategies. We will then introduce a novel framework, "Suspicious," which leverages dual auto-encoders to enhance the detection of anomalies, even in the presence of labeling errors in the training data. This chapter concludes with experimental evaluations, comparing the effectiveness of various methods on multiple datasets and demonstrating the resilience and efficiency of the proposed approach.

1.1 State of the Art

When considering graph data, the definition of what we will call an anomaly has evolved alongside the various models, tasks, and datasets available. Figure 1.1 illustrates the timeline presented in the survey on unsupervised graph anomaly detection [49]. However, one constant property defines this domain: the rarity of anomalies in the dataset induces a class imbalance.

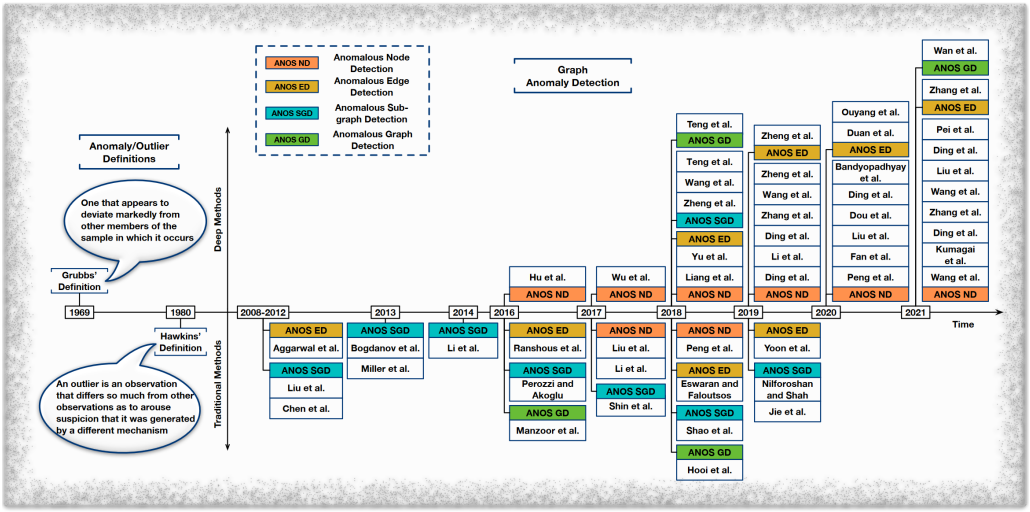


Figure 1.1: Timeline of Graph Anomaly Detection and Reviewed Techniques [49].

1.1.1 Graph Outlier Detection

The first methods to emerge were unsupervised ones designed exclusively for unattributed graphs, as most of the available datasets only contained relational data. An unattributed graph is defined as a graph $G = (V, \mathcal{E})$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, and $\mathcal{E} \subseteq V \times V$ is the set of edges represented by a symmetric adjacency matrix $\mathbf{A}$ where $a_{i,j} = 1$ if there is an edge between nodes i and j and $a_{i,j} = 0$ otherwise. Each edge $\mathcal{E}_{i,j}$ has an associated weight value $\mathcal{W}_{i,j} \in \mathbb{R}$.

At this point in time, anomalies were mostly referred to as outliers, and their definition was heavily influenced by expert knowledge specific to the application domain, primarily social networks.

SCAN

In SCAN [81], outliers are defined as:

> *Vertices that have only a weak association with a particular cluster of nodes.*

The method detects outliers by clustering nodes based on their common neighbors in the graph, assigning two nodes to the same cluster if they share many neighbors. This definition of outliers and the clustering method are both heavily influenced by the structures of social communities.

Although this method is efficient for those applications, it does not function well for other types of outliers or for attributed graphs. Subsequent efforts [56, 2] focused on extracting feature matrices from these graphs, allowing classical vectorial anomaly detection methods to be applied.

1.1.2 Network Representation

Following these initial methods, application cases diversified, necessitating the detection of outliers as they were defined in tabular data:

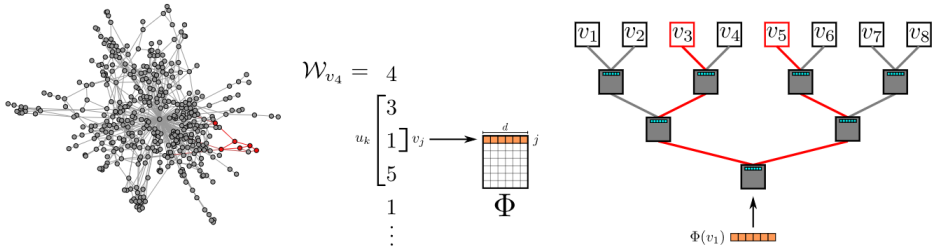
> *An outlier is an observation that differs so much from other observations as to arouse suspicion that a different mechanism generated it.* [31]

To detect these outliers, most state-of-the-art methods follow the same concept: using relational information to create a latent vectorial representation of the graph, reducing anomaly detection to a downstream task on

which vectorial anomaly detection methods can be applied.

DeepWalk

The DeepWalk [56] algorithm uses the SkipGram [51]. SkipGram is a language model that creates a latent word representation by maximizing the probability of co-occurrence among words that appear within a certain distance in a sentence. To adapt this model, which typically takes word sequences as input, to graph data, DeepWalk uses random walks from each node as the input sequence for SkipGram, as shown in Figure 1.2.



(a) Random walk generation. (b) Representation mapping. (c) Hierarchical Softmax.

Figure 1.2: Overview of DeepWalk [56].

Node2Vec

Node2Vec [28] improves DeepWalk by introducing a search bias α that allows the user to control the random path through two parameters p and q. These parameters control how quickly the walk explores and leaves the neighborhood of the starting node. Consider a random walk that just traversed the edge (t, v) and now resides at node v as illustrated in Figure 1.3. The walk needs to decide on the next step through edges (v, x) leading from v. The

unnormalized transition probability used to guide the choice is:

$$\alpha_{pq}(v,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases} \tag{1.1}$$

where d_{tx} denotes the shortest path distance between nodes t and $\mathbf{X}$. In this way, p controls the likelihood of immediately revisiting a node. A high p (i.e., $p > \max(q, 1)$) ensures that we are less likely to revisit a node, while a low p (i.e., $p < \min(q, 1)$) would lead the walk backward, keeping the walk near the original node. Similarly, a high q (i.e., $q > \max(p, 1)$) tends to keep the walk local around the original node, while a low q (i.e., $q < \min(p, 1)$) would lead the walk outward.

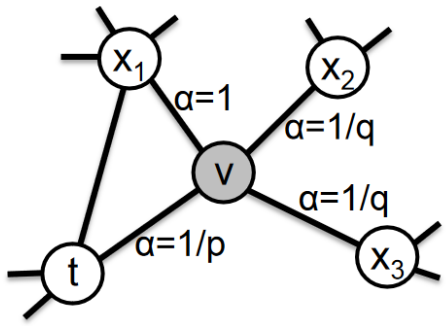


Figure 1.3: Illustration of the random walk procedure in Node2Vec. The walk transitions from node t to node v and evaluates its next step from node v. Edge labels indicate search biases α [28].

OddBall

OddBall [2] focuses on unattributed graphs. It proposes to take the ego-net of each node v_i, i.e., all nodes that are linked to v_i by an edge, to produce a feature vector whose components correspond to:

- $|\mathcal{N}(v_i)|$: the number of direct neighbors of v_i, i.e., nodes directly linked to v_i,
- $|\mathcal{E}_{\mathcal{N}(v_i)}|$: the number of edges in the ego-net,
- $\mathcal{W}_{v_i}$: the sum of all weights in the ego-net,
- $\lambda_{w,i}$: the principal eigenvalue of the weighted adjacency matrix in the ego-net.

The authors then use the Egonet Weight Power Law (EWPL), Egonet λ_w Power Law (ELWPL), Egonet Rank Power Law (ERPL), and Egonet Density Power Law (EDPL), a series of power laws that state each of these features is linked to the others by a power law. To compute an anomaly score that compares the egonet properties to the established power law, they define y as the value of one of these features of a node v_i (denoted as y_i), and similarly, let x_i denote the value of another of those features for node v_i. Given the power law equation $\hat{y}_i = Cx_i^\theta$ that links the particular pair of features $\mathbf{X}$ and y, they define the outlierness score of node v_i to be:

$$\text{out-line}(i) = \frac{\max\left(y_i, Cx_i^\theta\right)}{\min\left(y_i, Cx_i^\theta\right)} \cdot \log\left(|y_i - Cx_i^\theta| + 1\right) \tag{1.2}$$

The equation above measures the deviation of feature y_i observed for node v_i from its expected value given by the power law with regard to its feature x_i, giving us a score proportional to the outlierness of a node, with the minimum being 0.

In addition, they combine this score with the score obtained by using the Local Outlier Factor defined in Equation 1.5, on this feature vector to obtain a final anomaly score.

Takeaway

The development of network representation methods like DeepWalk and Node2Vec marked a significant advancement in graph mining. By leveraging relational information to create latent vectorial representations, these methods allowed traditional anomaly detection techniques to be applied to graphs.

1.1.3 Vectorial Anomaly Detection

Once the latent representations have been obtained, those vectorial representations are fed as input to methods designed for tabular anomaly detection such as local outlier factor.

Local Outlier Factor

The Local Outlier Factor (LOF) [11] is a vectorial density-based score that detects data points that are far from all other data points by defining the k-distance, Reachability Density (rd), and Local Reachability Density (lrd). The k-distance $kdist$ of an element is the distance of an element to its k^{th} nearest neighbor. The choice of the distance metric D depends on the specific problem at hand (Euclidean, Manhattan, etc.).

The reachability density between an element i and an element j is defined as:

$$rd(i, j) = \max(kdist(j), D(i, j)) \tag{1.3}$$

Where and $D(i, j)$ is a distance between i and j.

The local reachability density of an element i is defined as the inverse of

the average reachability distance to its neighbors:

$$\mathrm{lrd}(i) = \left(\frac{1}{|N_k(i)|} \sum_{j \in N_k(i)} rd(i,j) \right)^{-1} \tag{1.4}$$

where $N_k(i)$ is the set of its k nearest neighbors of i. This density measures how far a point is from the nearest cluster of points. Low values of lrd imply that the closest cluster is far from the point.

The LOF of an element i is the average ratio of the local reachability density of i to the local reachability density of its k-nearest neighbors:

$$\mathrm{LOF}(i) = \frac{\sum_{j \in N_k(i)} \mathrm{lrd}(j)}{|N_k(i)|} \times \frac{1}{\mathrm{lrd}(i)} \tag{1.5}$$

If the point is normal, the ratio of the average lrd of neighbors is approximately equal to the lrd of a point (because the density of a point and its neighbors are roughly equal), giving a LOF score roughly equal to 1. Generally, a LOF score above 1 is considered an outlier, but that is not always true.

Isolation Forest

Isolation Forest (IF) [43] is an ensemble-based unsupervised anomaly detection algorithm. It isolates anomalies by randomly splitting the data points along randomly chosen features. The number of splits required to isolate a data point is considered its anomaly score. Anomalies, by definition, tend to have fewer splits needed for isolation compared to regular data points due to their inherent deviation from the majority.

The combination of these vectorial methods applied to the latent representation produced by methods in the previous section has shown great performance for many tasks. However, they have since been outperformed on most of those tasks by methods utilizing new neural networks designed

specifically for relational data.

Takeaway

Vectorial anomaly detection methods like LOF and IF, when applied to latent representations of graph data, offered robust performance in many tasks. However, their reliance on traditional tabular data processing limited their effectiveness in dealing with the inherent relational properties of graphs that are not fully captured by those latent representations. This performance gap paved the way for adopting graph neural networks (GNNs) that are specifically designed to handle graph-structured data.

1.1.4 Graph Neural Networks (GNN)

Recently, Graph Neural Networks (GNNs), deep neural network architectures developed directly for tasks on relational data, have emerged based on the same general architecture:

General Architecture

Let $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ be an attributed network defined by the set of nodes $\mathcal{V} = \{v_1, \ldots, v_n\}$, the set of edges $\mathcal{E}$ represented by a symmetric adjacency matrix $\mathbf{A}$ where $a_{i,j} = 1$ if there is an edge between nodes i and j and $a_{i,j} = 0$ otherwise, and the attribute matrix $\mathbf{X} \in \mathbb{R}^{(n \times d)}$ which ith row $\mathbf{x}_i$ represents the attribute vector of v_i. In the same way, $\mathbf{a}_i$ denotes the ith row of A.

Most GNNs will follow the same two steps:

- Neighbor Aggregation: The first step is aggregating the information of a node with the information of its neighbors.
- Feature Update: After aggregating the neighbors' information, the model applies a multilayer perceptron (MLP) to learn patterns.

By stacking l layers of this architecture, the model can learn the representation of the l-hop neighborhood of each node.

A **Multilayer Perceptron** (MLP) [16] consists of an input layer, multiple hidden layers with interconnected neurons, and an output layer as shown in Figure 1.4. Information propagates forward, undergoing linear transformations and non-linear activation functions in each layer.

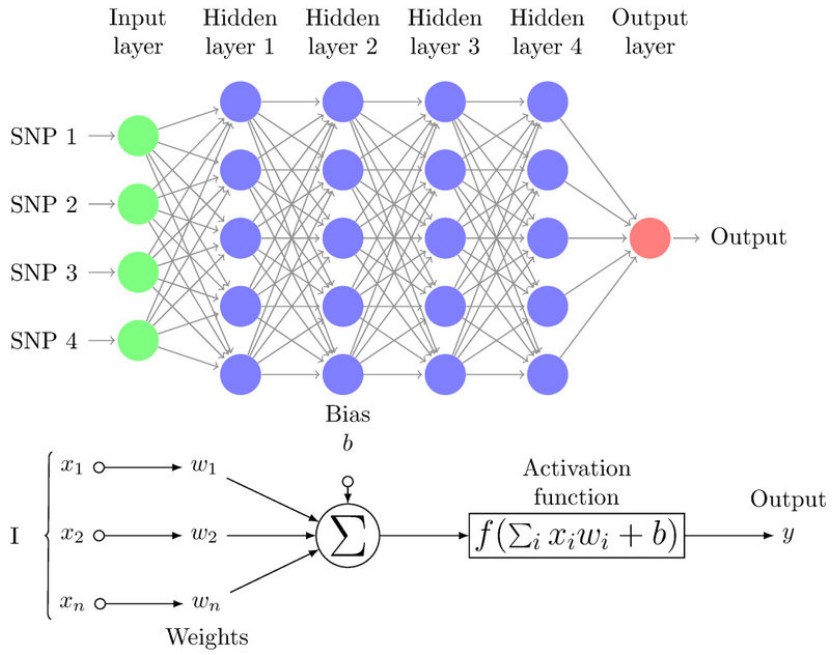


Figure 1.4: Multilayer Perceptron (MLP) diagram with four hidden layers. One neuron is the result of applying the nonlinear transformations of linear combinations (x_i, w_i, and biases b). [16]

Graph Convolutional Network

Graph Convolutional Network (GCN) [36] is one of the first GNNs to be developed. It proposes to aggregate the attributes of a node with the attributes

of its neighbors through a convolution. The core idea is to perform convolutions directly on the graph, leveraging its structure to propagate information between nodes. The graph convolutional layer can be defined as follows:

$$\mathbf{H}^{(l+1)} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) \tag{1.6}$$

where $\mathbf{H}^{(l)}$ is the latent representation of the layer l. $= \mathbf{A} + \mathbf{I}$ with $\mathbf{I}$ as the identity matrix, and $\tilde{\mathbf{D}}_{i,i} = \sum_j \tilde{\mathbf{A}}_{i,j}$, $\mathbf{W}^{(l)}$ is the learned weight matrix, and σ is an activation function, i.e., a function applied to each neuron's output to introduce non-linearity into the model. The initial input is $\mathbf{X}$, the attribute matrix.

$$\mathbf{H}^{(0)} = \mathbf{X} \tag{1.7}$$

The pooling method $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}$ essentially replaces the feature of a node with the mean feature in its own egonet.

GraphSAGE

GraphSAGE [30] is a framework for inductive representation learning on large graphs. Unlike traditional graph convolutional networks that aggregate the attributes of all neighboring nodes, GraphSAGE samples a fixed-size set of neighbors for each node and then aggregates their attributes through an aggregation function such as mean, LSTM, or pooling. Specifically, GraphSAGE performs the following steps, as shown in Figure 1.5, for each node in the graph:

- Neighbor Sampling: For each node, a fixed-size set of neighbors is sampled. This ensures computational efficiency and allows the method to scale to large graphs.
- Feature Aggregation: The sampled neighbors' attributes are aggregated using a neural network. The aggregation function can be mean, LSTM, or pooling-based.

- Embedding Update: The node's embedding is updated by combining its current embedding with the aggregated neighbor information.

Formally, the embedding H_i for a node v_i at layer $k+1$ is expressed as:

$$H_i^{k+1} = \sigma\left(W^k \cdot \text{AGGREGATE}\left(\left\{h_u^k, \forall u \in N(v_i)\right\}\right)\right) \tag{1.8}$$

where H_i^k is the embedding of node i at layer k, $N(v_i)$ is the set of neighbors sampled for node v_i, W^k is the weight matrix for layer k, σ is a non-linear activation function, such as ReLU, and AGGREGATE is the aggregation function.

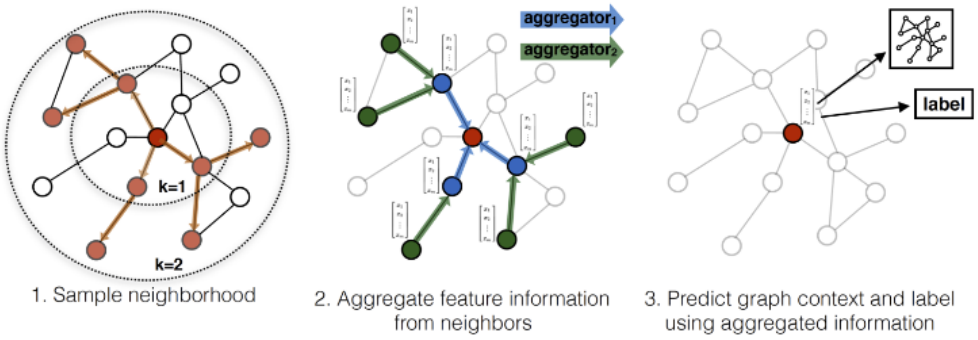


Figure 1.5: Visual illustration of the GraphSAGE sample and aggregate approach [30].

Graph Isomorphism Network

Graph Isomorphism Network (GIN) [80] is a powerful neural network architecture designed for graph representation learning. GIN is inspired by the Weisfeiler-Lehman (WL) graph isomorphism test, which is a method used to determine if two graphs are structurally identical. GIN leverages this concept to achieve better expressive power in distinguishing different graph structures.

The key components of GIN are as follows:

- Neighbor Aggregation: GIN aggregates the features of the neighbors of a node. Unlike traditional graph neural networks that use weighted sums or means, GIN uses sum aggregation, which theoretically provides maximum discriminative power.
- Feature Update: After aggregating the neighbors' features, GIN updates the node's features by applying an MLP. This process ensures that the node features are sufficiently transformed and can capture complex patterns.

Formally, the embedding H_i for a node v_i at layer $k+1$ is computed as:

$$H_i^{k+1} = \text{MLP}^{(k)}\left((1+\epsilon^{(k)}) \cdot H_i^{(k)} + \sum_{u \in \mathcal{N}(v_i)} H_u^{(k)}\right) \tag{1.9}$$

In this equation, $H_i^{(k)}$ is the embedding of node v_i at layer k, $\epsilon^{(k)}$ is a learnable or fixed scalar that allows the model to learn the importance of the node's own features, and $\text{MLP}^{(k)}$ is a multilayer perceptron applied at layer k.

Graph Attention Network

Graph Attention Networks (GAT) [74] introduce the attention mechanism to graph neural networks, allowing dynamic weights in the aggregation process based on the importance of neighboring nodes. GATs have been shown to be effective in various graph-based tasks due to their ability to assign different importance to different nodes in a neighborhood.

The core idea of GAT is to compute the attention coefficients, which determine the importance of a node's neighbors, as shown in Figure 1.6. This process involves the following steps:

- Attention Mechanism: GAT computes attention coefficients for each edge, indicating the importance of neighboring nodes. This is achieved

using a shared attention mechanism that calculates the coefficients based on the features of the nodes involved.

- Feature Aggregation: The node features are aggregated using the attention coefficients to produce a new feature representation for each node.

Formally, the node embedding $h_i^{(l+1)}$ for a node v_i at layer $l+1$ is computed as:

$$h_i^{(l+1)} = \sigma\left(\sum_{v_u \in \mathcal{N}(v_i)} \alpha_{iu}^{(l)} W^{(l)} h_u^{(l)}\right) \tag{1.10}$$

The attention coefficients $\alpha_{iu}^{(l)}$ are computed as:

$$\alpha_{iu}^{(l)} = \frac{\exp\left(\text{ReLU}\left(a^{(l)T}[W^{(l)}h_i^{(l)}||W^{(l)}h_u^{(l)}]\right)\right)}{\sum_{v_k \in \mathcal{N}(v_i)} \exp\left(\text{ReLU}\left(a^{(l)T}[W^{(l)}h_i^{(l)}||W^{(l)}h_k^{(l)}]\right)\right)} \tag{1.11}$$

In these equations, $h_i^{(l)}$ and $h_u^{(l)}$ represent the embeddings vectors of nodes v_i and v_u at layer l, $\alpha_{iu}^{(l)}$ is the attention coefficient indicating the importance of node v_u's features to node v_i, $a^{(l)}$ is a learnable weight vector used in computing the attention coefficients, $||$ denotes the concatenation operation, and ReLU is the activation function applied to the concatenated features.

Takeaway

The introduction of Graph Neural Networks (GNNs), including GCN, GraphSAGE, GIN, and GAT, changed the whole domain of graph mining, including anomaly detection on attributed networks. These models capture structural and attribute information through advanced aggregation and feature update mechanisms and are used as backbones by many techniques, creating the domain of deep graph anomaly detection.

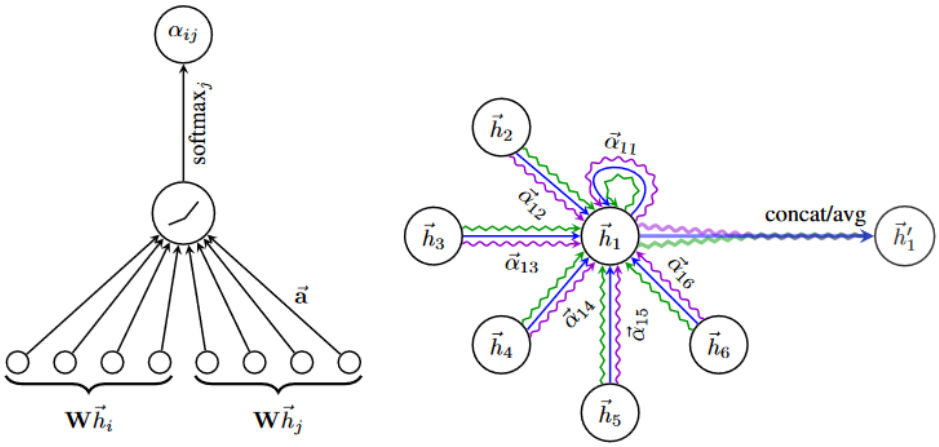


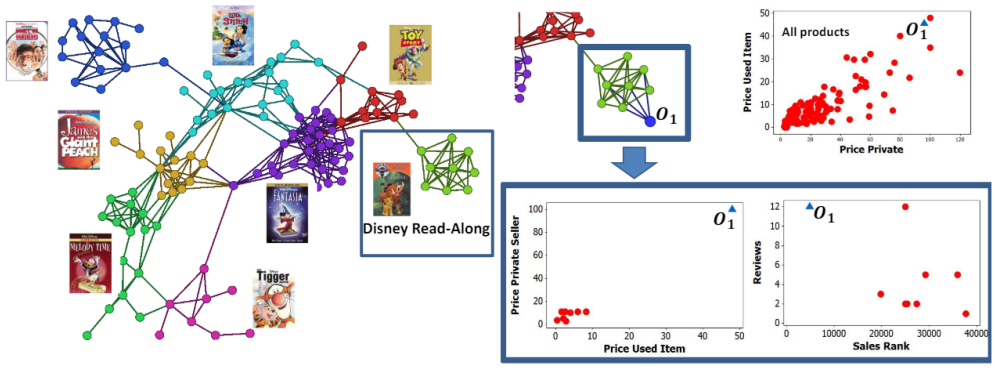
Figure 1.6: Left: The attention mechanism $a(W_{\mathbf{h}_i}, W_{\mathbf{h}_j})$ employed by the model, parametrized by a weight vector $\mathbf{a} \in \mathbb{R}^{2F}$, applying a ReLU activation. Right: An illustration of multihead attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\mathbf{h}'_1$ [74].

1.1.5 Graph Datasets for Anomaly Detection

The arrival of these new models coincides with the arrival of new relational datasets with labels representing real-world anomalies:

- **Disney** [54] is an Amazon co-purchase network that contains only Disney DVDs that are linked by an edge if they are often bought together (124 nodes with 334 edges). The 30 attributes per node are the extracted product information (e.g., product prices, different rating ratios, product reviews). The existing graph clusters correspond to similar Disney films, such as Disney Pixar films or Disney classics. The ground truth was obtained from a user experiment. Outliers have been labeled by a class of high school students as domain experts for the selected subgraph. Each co-purchased group was shown to the students as a product list, and they had to label one or two items that they con-

sidered deviating from the others in the group. For the ground truth, all products that have been labeled as outlying by at least 50% of the students are considered outliers. The dataset is illustrated in Figure 1.7.



(a) Subgraph of Disney DVD products

(b) Different subspaces of the given attributes

Figure 1.7: An outlier example in a subgraph of the Amazon co-purchased network [54].

- **Tencent-Weibo** dataset [88] is a user-posts-hashtag graph from a Chinese social media platform, i.e., users, posts, and hashtags are the nodes. A user is linked by an edge to the posts they participated in. Posts are linked by an edge to the hashtags they contain. It includes 8,405 users and 61,964 hashtags. Users were labeled based on temporal information. The algorithm assumed that posting two messages within specific intervals, such as 10, 15, 30, 45, and 60 seconds, constitutes a suspicious event. Users who made at least 5 suspicious events were labeled as suspicious users, while those with no suspicious events were labeled as benign users. Consequently, there are 868 suspicious users and 7,537 benign users. Since the ground truth was derived from time information, timestamps were not used to create raw user features. Thus, the raw feature vector comprises two parts: a one-hot vector

representing user posts and a bag-of-words feature.

- **Reddit** [39] is a graph representing the relationship between users and subreddits on the social media platform, Reddit. Users and subreddits are nodes, and there is a link between a user and a subreddit if the user participates in it. This publicly available dataset includes one month of user posts on subreddits. The dataset comprises the 1,000 most active subreddits and the 10,000 most active users, represented as subreddit nodes and user nodes, respectively. This results in a total of 168,016 interactions. Each user is assigned a binary label indicating whether they have been banned by the platform. Banned users are considered anomalies compared to regular Reddit users. The text of each post is transformed into a feature vector, and the features of users and subreddits are obtained by summing the features of their respective posts.

- **Books** [63] is an Amazon co-purchase network, where each node is a book sold by Amazon, and two products are linked by an edge if they are often bought together. The feature vectors are derived from the product information (21 attributes). In this dataset, the labels were acquired through an 'Amazon fail' tag, a system that enabled all Amazon users to report issues. It is evident that the reasons for this tag can be highly diverse, as there are no specific guidelines provided to users on when to use it. A book is deemed anomalous if it has been tagged by 20 people.

- **DGraph** [34] is a large-scale attributed graph containing 3 million nodes, 4 million dynamic edges, and 1 million ground truth nodes. The nodes correspond to user accounts in a financial company that offers personal loan services, and an edge between two nodes indicates that one account has added another as an emergency contact. For accounts with at least one borrowing record, anomalies are those with overdue

histories, while normal nodes are those without. Additionally, there are 2 million accounts/nodes with no borrowing records at all. The 17 node features are derived from user profile information such as age and gender.

In those cases, an anomaly is simply what human operators decide is not normal. That means that there can be many definitions of what constitutes an anomaly, and there is no guarantee that two experts will always agree that the same pattern is anomalous. This makes the task much more complex, as, unlike previous understandings, no predefined pattern can be found.

To expand their benchmarks, a new protocol to inject anomalies in a real-world dataset was formalized by [45]. These anomalies are defined as:

- **Contextual anomalies**: To inject a total of o contextual outliers, first, sample o nodes from the node-set V without replacement; these are the nodes whose attributes we aim to modify to turn them into contextual outliers. We denote the set of these o nodes as V_c (so that $o = |V_c|$), and refer to the remaining nodes $V_r = V \setminus V_c$ as the "reference" set. For each node $i \in V_c$, q nodes are chosen at random from the reference set V_r. Among these q reference nodes chosen, we find the one whose attributes deviate the most (in terms of Euclidean distance) from those of node i. We then change the attributes of node i to be the same as those of this most dissimilar reference node found.

- **Structural anomalies**: The basic strategy is to create n non-overlapping densely connected groups of nodes, where each group has exactly m nodes (so that there are a total of $m \times n$ structural outliers injected). To do this, for each $i = 1, \ldots, n$, we randomly sample m nodes to form the i-th group (these m nodes are sampled uniformly at random from nodes that have not been previously chosen to form a group); for these m nodes, we first make them fully connected and then drop each edge independently with probability p.

The injected anomalies presented are similar to the outliers that were already identified, that is, nodes different from their neighbors. This correspond to the homophily property, which describes a graph where nodes with similar features or the same class labels are likely to be neighbors, i.e., nodes linked together. Contrary to heterophily which occurs when nodes have dissimilar features and different class labels from their neighbors. However, many real-world anomaly datasets do not fit that description, meaning the homophily property might have been given disproportionate importance in the previous methods.

Takeaway

With these new datasets, the notion of an outlier as previously defined does not encompass the anomalies presented, leading to a new definition of anomaly based on the notion of normality, a vague concept that changes depending on the application case.

> *Anomalies are patterns in data that do not conform to a well-defined notion of normal.* [13]

1.1.6 Deep Graph Anomaly Detection

With the advent of GNNs and the appearance of those new datasets, we saw a substantial increase in publications for the graph anomaly task, as seen in Figure 1.1 from Section 1.1. In the following sections, we will now focus only on the methods applied for the node anomaly detection task on the attributed network where the anomalous element that needs to be found is a node.

1.1.7 Unsupervised Anomaly Detection

Currently, the most efficient unsupervised anomaly detection methods are led by reconstruction-based techniques [45] that aim to exploit the structure of the graph and the attributes of the nodes to detect anomalies [41, 55, 17, 19, 8]. They create an approximation of the original graph and then consider the distance of each node to its reconstruction as an anomaly score.

Anomalous

Anomalous [55] is based on the CUR decomposition, which states that for any matrix $\mathbf{M}$, there exist three matrices:

- $C \in \mathbb{R}^{d \times m}$: A submatrix containing m columns of M,
- $R \in \mathbb{R}^{r \times n}$: A submatrix containing r rows of M,
- $U \in \mathbb{R}^{m \times r}$: A small matrix that mathematically combines the information from $\mathbf{C}$ and $\mathbf{R}$,

such that the product of these three matrices CUR is an approximation of M. In Anomalous, The authors use the structure of the graph to select the rows and columns of $\mathbf{X}$ that will be used in $\mathbf{C}$ and $\mathbf{R}$. Once the approximation of $\mathbf{X}$ is obtained, they apply residual analysis, which aims to study the distance between true data and estimated data to spot anomalies since anomalies usually have large residual errors caused by significant deviations from most reference instances in patterns [73].

In this category, Graph Auto Encoders (GAE) have gained popularity in recent years due to their performance [45, 49, 36, 8]. These models use GCN to create an encoder that generates a low-dimensional vector representation of the nodes using both the attribute matrix and the adjacency matrix, before feeding this representation to another GCN to reconstruct the original graph from the latent representation.

Dominant

Dominant [17] is one of the first works that leverages GCN and AE for detecting outlier nodes. It uses a 3-layer encoder to encode both structural and attribute information into a single latent representation that is then used by two decoders to reconstruct the attribute matrix and the adjacency matrix, respectively, as shown in Figure 1.8.

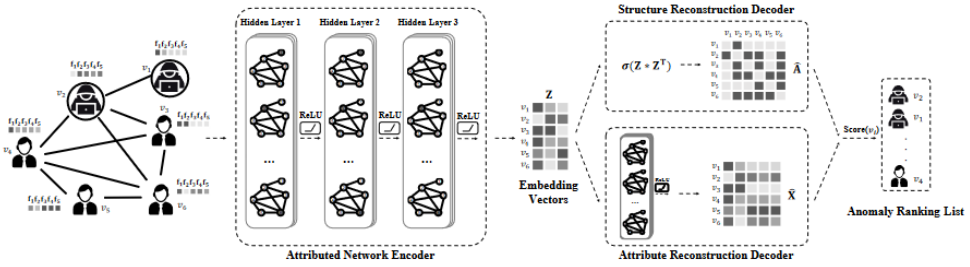


Figure 1.8: The overall framework of Dominant for deep anomaly detection on attributed networks [17].

Encoder: The attributed network auto-encoder 3-layer GCN uses ReLu as the activation function, giving us the following attribute encoder:

$$H^{(1)} = \text{Relu}\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}XW^{(0)}\right) \tag{1.12}$$

$$H^{(2)} = \text{Relu}\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(1)}W^{(1)}\right) \tag{1.13}$$

$$Z = H^{(3)} = \text{Relu}\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(2)}W^{(2)}\right) \tag{1.14}$$

After applying three layers of convolution, the input attributed network can be transferred to the latent representations Z.

Attribute Decoder: The attribute reconstruction decoder leverages another graph convolutional layer to predict the original nodal attributes as follows:

$$\hat{X} = \text{Relu}\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}ZW^{(3)}\right) \tag{1.15}$$

Structure Decoder: The decoder takes the latent representations as input and predicts whether there is a link between each pair of nodes:

$$\hat{A} = \text{sigmoid}(ZZ^T) \tag{1.16}$$

Anomaly Score: To jointly learn the reconstruction errors, the loss function to minimize includes both structural and node attribute errors:

$$L = (1-\alpha)R_S + \alpha R_A = (1-\alpha)\|A - \hat{A}\|_F^2 + \alpha\|X - \hat{X}\|_F^2 \tag{1.17}$$

where α is a balance parameter to control the relative importance of attribute information over structural information, $\|\cdot\|_F$ denotes the Frobenius norm of a matrix, and $\|\cdot\|^2$ the l_2 norm of a vector. Finally, the reconstruction error is computed from both higher-order structures and node attribute perspectives to calculate the anomaly score of each node:

$$\text{score}(v_i) = (1-\alpha)\|a_i - \hat{a}_i\|_F^2 + \alpha\|x_i - \hat{x}_i\|_F^2 \tag{1.18}$$

giving a score such that a higher value indicates a higher probability of being an abnormal node.

AnomalyDAE

In AnomalyDAE [19], the authors use the same architecture as Dominant while replacing GCN with GAT layers and adding two hyper-parameters to the loss, θ and η, which are masks that allow the user to impose a greater penalty on the reconstruction error of the non-zero elements. This loss function L is defined by:

$$L = \alpha\|(A - \hat{A}) \odot \theta\|_F^2 + (1-\alpha)\|(X - \hat{X}) \odot \eta\|_F^2 \tag{1.19}$$

where $\odot$ denotes the Hadamard product, and α is a trade-off parameter.

GUIDE

In GUIDE [86], the authors propose an architecture with two auto-encoders, one for attributes and one for structure, as illustrated in Figure 1.9. Only the attribute encoder is based on the GCN architecture presented in Dominant, while the structure encoder is based on a GAT architecture.

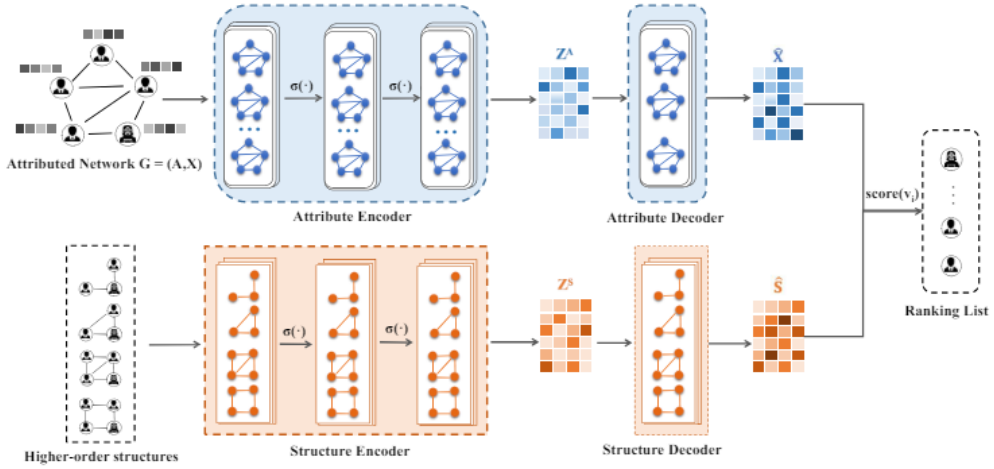


Figure 1.9: The framework of GUIDE [86].

Attribute Auto-encoder: The attribute auto-encoder network layers are the same as Dominant, with a 2-layer encoder that creates Z^A, the attribute latent representation, that is only used by the 1-layer attribute decoder to produce $\hat{X}$:

$$\hat{X} = \text{Relu}\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}Z^A W^{(3)}\right) \tag{1.20}$$

Structure Auto-encoder: Unlike Dominant, GUIDE's structural encoder uses a GAT-based autoencoder, following the equations presented in 1.1.4. It consists of a 2-layer encoder that generates Z^S, the structural latent representation, which is then fed into a 1-layer decoder to produce $\hat{S}$, the reconstructed structure.

$$\hat{S} = graph\_decoder(Z^S) \tag{1.21}$$

Once both reconstruction errors have been obtained, they are used in the same way as the reconstruction errors presented in Dominant.

However, recent benchmarks [45] have shown that while these methods perform well on datasets with injected anomalies, they often perform poorly on real-world anomaly datasets. This limitation arises from the rigidity in how unsupervised methods define "normal" behavior, which is often based solely on patterns learned from the majority of the data. In real-world scenarios, normal behavior can be more diverse and extremely context-dependent. This observation could explain the trend from unsupervised to semi-supervised/supervised methods in recent years.

Contrastive Learning for Anomaly Detection in Graphs

CoLA [47] addresses the anomaly detection problem in large graphs through the use of contrastive learning. Contrastive learning is a self-supervised learning approach where a model learns to differentiate between similar and dissimilar data points without explicit labels. Cola specifically relies on creating "node vs local neighborhood" pairs. A positive pair is created from a node embedding, and the subgraph embedding is from the subgraph that contains that node. In contrast, a negative pair is composed of a node embedding with a subgraph embedding that does not contain the node. This method is entirely built on the hypothesis that anomaly nodes have a mismatch between their near neighbors.

Subgraph sampling : For each node, a local subgraph is defined by a random walk starting from the target node. In this local subgraph, the attribute vector of the target node is replaced by a zero vector to hide it from the later contrastive learning. Then, for each target node, both a positive pair is created by taking this subgraph, and a negative pair is created by taking another node subgraph. The pairs are then added to a corresponding

sample pool, as shown in Figure 1.10. The corresponding instance pair P_i is denoted as:

$$P_i = (v_i, G_i, y_i) \tag{1.22}$$

where v_i is the target node, G_i is the local subgraph containing n_i and y_i the pair label with $y_i = 1$ if the pair is positive and $y_i = 0$ if the pair is negative.

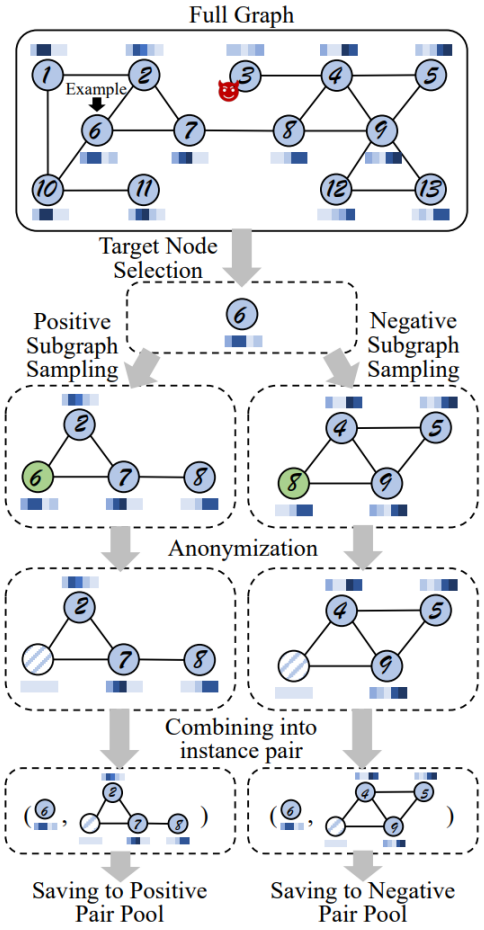


Figure 1.10: The sampling process of contrastive instance pairs. Here, the node v_6 is selected as the example of the target node. The initial nodes for subgraph sampling are marked in green. The blue-white stripe means the embedding of the corresponding node is masked with a zero vector. Figure from [47].

Then, as shown in Figure 1.11, the learning model is split into 3 parts: a GNN embedding module, a readout module, and a discriminator module.

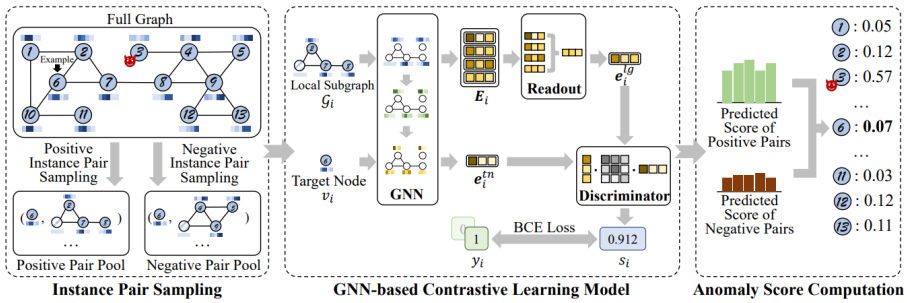


Figure 1.11: The overall framework of CoLA. The framework is composed of three components: instance pair sampling, GNN-based contrastive learning model, and anomaly score computation. Node v_3 is an anomaly since it has corrupted attributes, and the rest of the nodes are normal nodes. Figure from [47].

The GNN embedding module functions in the exact same way as the encoder seen in 1.1.7 to create e_i^{tn}, the target node embedding of the node v_i. Cola then uses these node embeddings to create a subgraph embedding vector e_i^{lg} for the local subgraph G_i with an average pooling function:

$$e_i^{lg} = \text{Readout}(E_i) = \frac{1}{n_i}\sum_{k=1}^{n_i}(e_k^{tn}) \tag{1.23}$$

Finally, the Discriminator, the contrastive part of the learning model, contrasts the embedding of the local subgraph e_i^{lg} with the embedding of the target node e_i^{tn} given by the encoder and outputs the pair predicted score by applying a bilinear scoring function:

$$s_i = \text{Discriminator}(e_i^{lg}, e_i^{tn}) = \sigma\left(e_i^{lg} W e_i^{tn\top}\right) \tag{1.24}$$

Put together, these three modules produce a binary classification model to predict the labels of contrastive instance pairs, trained with a binary cross

entropy loss:

$$L = -\sum_{i=1}^{N} [y_i \log(s_i) + (1 - y_i) \log(1 - s_i)] \tag{1.25}$$

Finally, an anomaly score is computed by using both $s_i^{(+)}$ and $s_i^{(-)}$ respectively the positive and negative pair scores:

$$f(v_i) = s_i^{(-)} - s_i^{(+)} \tag{1.26}$$

Takeaway

GAE-based techniques have proven effective and seen many iterations such as Dominant, AnomalyDAE, and GUIDE. These iterations show minor progression but keep a relative lead over other unsupervised techniques [45]. Cola differentiated itself by using contrastive learning, reaching good performance on injected anomalies but performing particularly poorly on real-world datasets [47, 45]. These limitations have led to a growing interest in exploring semi-supervised approaches that can leverage even limited labeled data to improve detection accuracy.

1.1.8 Semi-Supervised Anomaly Detection

While unsupervised methods have proven effective, they often struggle with real-world anomalies. To address this, recent approaches have shifted toward semi-supervised methods, which leverage limited labeled data to improve detection accuracy. By combining labeled and unlabeled data, these methods enhance performance, particularly in complex anomaly scenarios.

Many models fully dedicated to anomaly detection have emerged, such as those presented in the following section.

Beta Wavelet Graph Neural Network

The Beta Wavelet Graph Neural Network (BWGNN) [72] is a backbone GNN designed to address the problem of anomaly detection in attributed networks. The authors identify a 'right-shift' phenomenon, where the spectral energy distribution of a graph with anomalies shifts towards higher frequencies. This motivates the need for spectral localized band-pass filters to detect anomalies effectively. The BWGNN starts with an original wavelet ψ and employs a group of wavelets as bases $W = (W_{\psi_1}, W_{\psi_2}, ...)$ in a Beta wavelet transform applied on a graph signal x, the attribute matrix X in our context, this transformation can be defined as:

$$W_{\psi_i}(x) = U g_i(\Lambda) U^T x \tag{1.27}$$

where $g_i(\cdot)$ is a kernel function in the spectral domain, and U and Λ are the eigenvectors and eigenvalues of the graph Laplacian L. The beta wavelet transform is expressed as:

$$W_{p,q} = U\beta^*_{p,q}(\Lambda)U^T = \beta^*_{p,q}(L) = \frac{(L/2)^p(I - L/2)^q}{2B(p+1, q+1)} \tag{1.28}$$

where $\beta^*_{p,q}(w)$ is a Beta distribution, p and q are parameters controlling the wavelet shape, and $B(p+1, q+1)$ is the Beta function. This transform ensures that the BWGNN can capture high-frequency anomalies through flexible, localized, and band-pass filters.

Semi-supervised GNN

In this contribution [38], the authors propose a method for semi-supervised anomaly detection, leveraging the Graph Convolutional Network (GCN) encoder architecture. However, unlike traditional approaches, they focus on identifying anomalies directly within the latent space generated by the encoder.

The anomaly score for each node is defined as the squared Euclidean distance between the node's embedding and a center c, which represents the average of the embeddings of the normally labeled nodes from the training set:

$$a(v_i) = \|h_i - c\|^2, \tag{1.29}$$

where h_i is the embedding of node v_i.

The objective function they introduce consists of two main components:

- **Mean Squared Distance of Normal Nodes**: This term minimizes the average squared distance between the embeddings of normal nodes and the center, effectively clustering these nodes around the center in the latent space:

$$L_{\text{nor}}(\theta) = \frac{1}{|V_n|} \sum_{n \in V_n} \|h_n - c\|^2, \tag{1.30}$$

  where V_n is the set of nodes labeled as normal.

- **AUC Regularization Term**: This term is a differential approximation of the Area Under the Curve (AUC) and ensures that the embeddings of anomalous nodes are far from the center. The function $sigmoid(x)$ is the sigmoid function, which helps in differentiating the anomalies from the normal nodes:

$$R_{\text{AUC}}(\theta) = \frac{1}{|V_a||V_n|} \sum_{n \in V_a} \sum_{m \in V_n} sigmoid(a(v_n) - a(v_m)), \tag{1.31}$$

  where V_a is the set of nodes labeled as anomalous.

The final objective function to be minimized is a combination of these two terms:

$$L(\theta) = L_{\text{nor}}(\theta) - \lambda R_{\text{AUC}}(\theta), \tag{1.32}$$

Where λ is a hyperparameter that controls the influence of the AUC loss. By including the AUC regularizer, this method is able to learn more sophisticated node embeddings, improving its ability to detect anomalies accurately.

These methods actively call themselves anomaly detection methods. However, many supervised works prefer the term fraud detection. Although some works [18, 40, 18] use both terms interchangeably, many consider a different task with a stronger emphasis on identifying anomalous patterns.

In this work, we use the following fraud definition:

Frauds are the result of intentional illegitimate behavior or actions.

If we consider legitimate actions to be one of the possible notions of normality in our last definition of an anomaly, we can also consider that frauds are a subset of anomalies.

Fraud Detection

A significant number of works [40, 46, 18, 24, 67] take on a more application-based form, using graphs with heterogeneous edges. A heterogeneous graph contains either several types of entities as nodes or several types of relations as edges, contrary to a homogeneous graph contains only one type of entity as nodes and only one type of relation as edges. However these studies are generally limiting their experimentation to only two of those real-world graphs:

- **Amazon Fraud** [50] is a dataset that includes product reviews from the Musical Instruments category. This dataset is used for fraud detection tasks involving user reviews. The nodes in this dataset are individual users, each represented by 25 handcrafted features. The label is given by Amazon and represents fake reviews. The dataset includes three types of relationships between users:

  - U-P-U: This relation connects users who have reviewed at least one common product.
  - U-S-U: This relation connects users who have given at least one common star rating within one week.
  - U-V-U: This relation connects users who have the top 5% mutual review text similarities (measured by TF-IDF) among all users.

  The dataset contains a total of 11,944 nodes (users), with 6.87% labeled as fraudulent. It has 351,216 edges for the U-P-U relation, 7,132,958 edges for the U-S-U relation, and 2,073,474 edges for the U-V-U relation.

- **YelpChi** [60] consists of hotel and restaurant reviews that have been filtered (spam) and recommended (legitimate) by Yelp. This dataset is used to conduct spam review detection tasks. The nodes in this dataset are individual reviews, each represented by 32 handcrafted features. The dataset includes three types of relationships between reviews:
  - R-U-R: This relation connects reviews posted by the same user.
  - R-S-R: This relation connects reviews under the same product (hotel or restaurant) that have the same star rating.
  - R-T-R: This relation connects reviews under the same product that were posted in the same month.

  The dataset contains a total of 45,954 nodes (reviews), with 14.53% labeled as fraudulent (spam). It has 98,630 edges for the R-U-R relation, 1,147,232 edges for the R-T-R relation, and 6,805,486 edges for the R-S-R relation.

Most of these methods use a node classifier architecture [77, 32, 58, 46, 18, 24, 67], using a classical cross-entropy loss to train their GNN model:

$$L = -\sum_{i=1}^{N} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)] \tag{1.33}$$

where y_i is the true label of sample i (0 or 1), and p_i is the predicted probability of sample i being in class 1.

Most methods focus on trying to learn a representation in ways that allow the models to understand complex relationships [46, 18, 24, 67].

Pick and Choose GNN

Pick and Choose GNN (PC-GNN) [46] uses a label-balanced sampler to pick nodes and edges for sub-graph training using the sampling probability:

$$P(v) \propto \frac{\|\hat{A}(:, v_i)\|_2}{LF(C(v_i))} \tag{1.34}$$

where $LF(C(v_i))$ is the label frequency of the class of v_i, $\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$ and $\hat{A}(:, v_i)$ is the column v_i of $\hat{A}$.

It then uses a neighborhood sampler that over-samples the neighborhood of the fraud class and under-samples the neighborhood of the normal class. This structure is then fed to a classical GNN classifier, such as the one described in Section 1.1.4, to obtain a final binary classification as either fraud or benign, as illustrated in Figure 1.12.

H2-FDetector

H2-FDetector [67], illustrated in Figure 1.13, includes multilayer convolution and each layer adds two components:

- Connection identification: A one-layer MLP m is applied on each edge (u,v) to determine whether they are homophilic or heterophilic. m

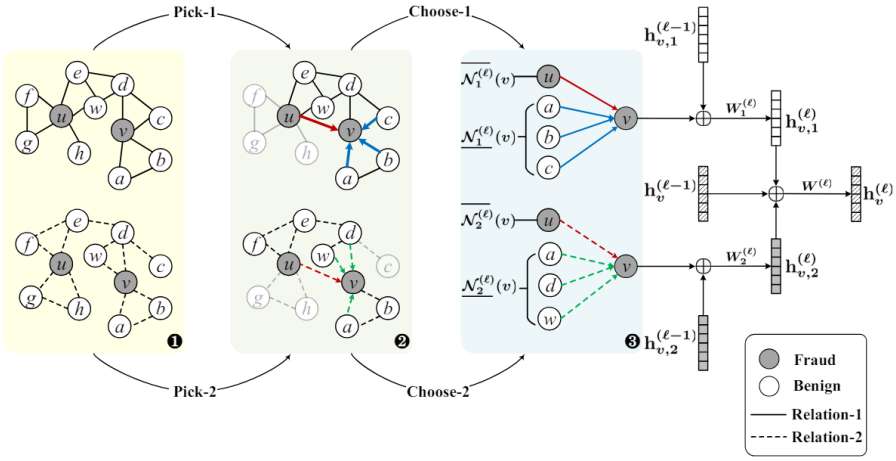


Figure 1.12: The figure demonstrates the l-th layer of PC-GNN framework on an example graph. The solid and dashed lines represent two kinds of relations among these nodes. The nodes in gray are fraudulent ones, and the white ones are benign. Figure from [46]

takes the embedding of both nodes after the convolution at the layer l as input:

$$m_{uv}^{(l)} = \tanh(W_c^{(l)}[\bar{h}_u^{(l)} \| \bar{h}_v^{(l)} \| (\bar{h}_u^{(l)} - \bar{h}_v^{(l)})]) \tag{1.35}$$

According to [10], the sum of neighborhood representations makes the representations similar, which is suitable for homophilic connections. At the same time, the difference between node features and neighborhood features makes the representations become discriminative, which suits heterophilic connections. Once the MLP has classified the edge, they determine c_{uv}. If the nodes are homophilic $c_{uv} = 1$, if they are heterophilic $c_{uv} = -1$:

$$c_{uv}^{(l)} = \text{SIGN}(m_{uv}^{(l)}) \tag{1.36}$$

- Connection aggregation: Follows the classical attention mechanism

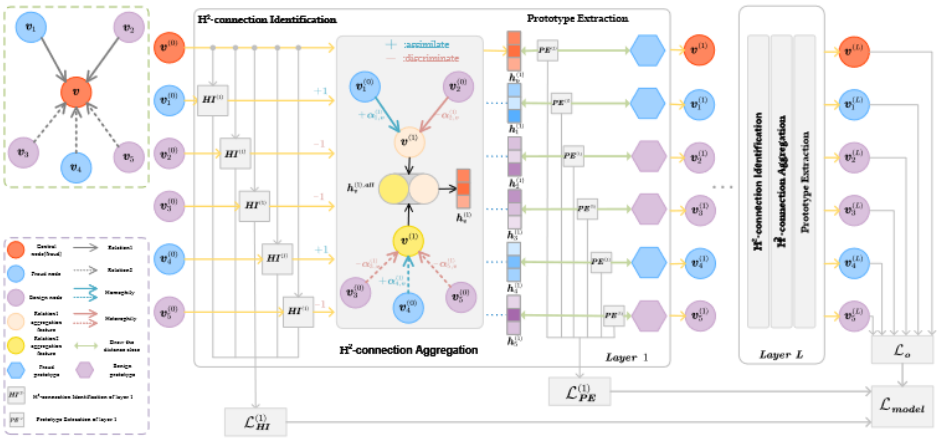


Figure 1.13: The Aggregation Process of Proposed H2-FDetector at the Training Phase. Figure from [67]

shown in GAT while adding c_{uv} in the attention computation:

$$\alpha_{vu}^{(l)} = \frac{\exp\left(\text{ReLU}\left(a^{(l)T}[W^{(l)}h_v^{(l)}||c_{uv}^{(l)}W^{(l)}h_u^{(l)}]\right)\right)}{\sum_{k\in\mathcal{N}(v_v)}\exp\left(\text{ReLU}\left(a^{(l)T}[W^{(l)}h_v^{(l)}||c_{uv}^{(l)}W^{(l)}h_k^{(l)}]\right)\right)} \tag{1.37}$$

- Final Classification: After multiple layers of aggregation, the model generates a final node embedding for each node. These embeddings are then fed in a last GAT layer used as a binary classifier. The model's ability to distinguish between homophilic and heterophilic neighbors allows it to adapt to the underlying graph structure more effectively, improving classification performance on both types of graphs.

Takeaway

Contrary to the previous sections, with the exception of some rare works such as GAD-Bench [71], which compare these models with a single methodology, these works tend to remain isolated in this one setup while not comparing

themselves with other anomaly detection methods, making it difficult to perfectly know the current state of general anomaly detection.

1.2 Suspicious

Most of these works do not take into account the real constraints that come with the task, such as:

- The rarity of labeled data. Identifying anomalies is a complex and time-consuming task for humans, making the production of the labeled datasets a costly endeavor.
- The reliability of labeled data. For the same reason as listed above, the humans producing those labels are subject to both biases and can make mistakes.

In practice, most of these methods need a large part of the dataset to be labeled and do not take into account the presence of mislabeled nodes (i.e., normal nodes labeled as frauds and conversely) within the training dataset that would seriously degrade the quality of classification between normal and fraudulent nodes.

To overcome this, we propose *Suspicious*[25], a semi-supervised methodological framework based on GAEs for identifying anomalies in an attributed graph, requiring only a small labeled sample and resilient to the labeling errors of the training sample.

1.2.1 Problem Formalization

Let $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ be an attributed network defined by the set of nodes $\mathcal{V} = \{v_1, \ldots, v_n\}$, the set of edges $\mathcal{E}$ represented by a symmetric adjacency matrix $\mathbf{A}$ where $a_{i,j} = 1$ if there is an edge between nodes i and j and $a_{i,j} = 0$

otherwise, and the attribute matrix $\mathbf{X} \in \mathbb{R}^{(n \times d)}$ which ith row $\mathbf{x}_i$ represents the attribute vector of v_i. In the same way, $\mathbf{a}_i$ denotes the ith row of A. It is assumed that there is a labeled subset of nodes, $\mathcal{V}_l \subset \mathcal{V}$. It is itself composed of two disjoint subsets, $\mathcal{V}_s$ and $\mathcal{V}_n$, containing nodes identified, respectively, as fraudulent and normal, possibly with mislabeling errors: $\mathcal{V}_l = \mathcal{V}_s \cup \mathcal{V}_n$ and $\mathcal{V}_s \cap \mathcal{V}_n = \emptyset$. This mislabeling, relatively frequent in practice, even when the labeling is done by experts, leads to anomalies being present in $\mathcal{V}_n$ and normal nodes in $\mathcal{V}_s$. We denote $\mathcal{V}_a \subset \mathcal{V}$ the set of the nodes that are truly anomalous, *i.e.*, the ground truth that is not available in real applications.

The problem we seek to solve can be expressed as follows: Given the attributed network $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ and the two labeled subsets $\mathcal{V}_s$ and $\mathcal{V}_n$, the goal is to compute an anomaly score for the unlabeled nodes such that the true fraudulent nodes have a higher score than the true normal ones.

1.2.2 Principle Behind Suspicious

To solve this task, *Suspicious* uses two graph auto-encoders, *Susp* and *Norm*, as shown in Figure 1.14. The auto-encoder *Norm* tries to reconstruct G such that the normal nodes are better reconstructed than the fraudulent ones, while *Susp* does the opposite and tries to reconstruct G in a way that anomalies are better reconstructed than normal nodes. For each unlabeled node, we then obtain a reconstruction error (*i.e.*, score) from *Norm* and another one from *Susp*, and, finally, these errors are combined in a ranking score. The formal definition of this score is given in the following section, but, informally, these two scores categorize the nodes as follows:

- A low reconstruction error by *Norm* and a high one by *Susp*: both models agree that this is a normal node which should result in a final score among the lowest.

- A high reconstruction error by *Norm* and a low one in *Susp*: both models agree that the node is fraudulent. This means that the node

corresponds to a relevant fraud and, therefore, it should result in a final score among the highest.

- In other cases, the combination of the two scores allows our model to better identify the anomalies belonging to $\mathcal{V}_s$ than each auto-encoder individually, as confirmed by the ablation study presented in Section 1.3.5.

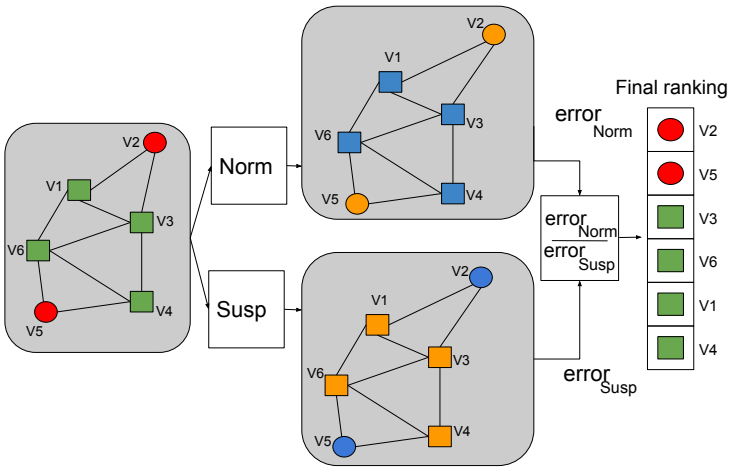


Figure 1.14: Architecture of Suspicious. *Norm* and *Susp* are auto-encoders similar to those shown in Figure 1.15. Green nodes are normal, red nodes are anomalous, orange nodes are poorly reconstructed, and blue nodes are well-reconstructed.

1.2.3 Architecture of Suspicious

The architecture of both auto-encoders used in *Suspicious* is illustrated in Figure 1.15 in which the colors correspond to the types of nodes: green for normal nodes, red for anomalies, orange for poorly reconstructed nodes, and blue for well-reconstructed nodes. The encoder (ENC in Equation 1.38) is a GCN [36] that computes an embedding $\mathbf{Z}$ of the nodes in G. Then the attribute decoder DEC (another GCN) and the adjacency matrix decoder

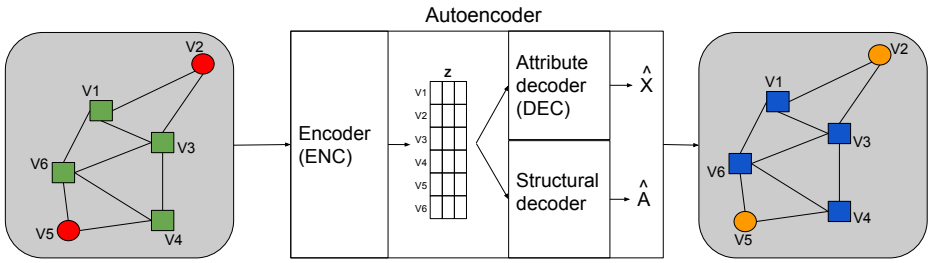


Figure 1.15: Architecture of an auto-encoder. Green nodes are normal, red nodes are anomalies, orange nodes are poorly reconstructed, and blue nodes are well-reconstructed.

are used to recreate, from $\mathbf{Z}$, the approximations $\hat{\mathbf{X}}$ and $\hat{\mathbf{A}}$ of $\mathbf{X}$ and $\mathbf{A}$, respectively:

$$\mathbf{Z} = \text{ENC}(\mathbf{A}, \mathbf{X}, \Theta_1); \hat{\mathbf{X}} = \text{DEC}(\mathbf{A}, \mathbf{Z}, \Theta_2); \hat{\mathbf{A}} = (\mathbf{Z}^t\mathbf{Z}). \tag{1.38}$$

Both auto-encoders, *Susp* and *Norm*, use this architecture with different learnable parameters sets: Θ_1^s and Θ_2^s for *Susp* and Θ_1^n and Θ_2^n for *Norm*.

We can notice that each auto-encoder has the same architecture as Dominant [17], but our model uses two of them and does not use the same loss function as Dominant, as explained in the next section. Furthermore, our framework is not limited to GCNs and can be used with other graph embedding models. We also implemented versions[1] of our framework in which ENC and DEC are GraphSAGE [30] or SGC [78] instead of the GCN.

1.2.4 Calculation of Reconstruction Errors

For each node v_i and each auto-encoder AE, which can be either *Susp* or *Norm*, we define the node reconstruction error as:

[1]Codes available on https://src.koda.cnrs.fr/labhc/code4publications/2023-ICTAI-suspicious

$$err_{AE}(v_i) = (1-\alpha)\|\mathbf{a}_i - \hat{\mathbf{a}}_i\|^2 + \alpha\|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2. \tag{1.39}$$

where α is a parameter that allows one to weight the importance given to each type of reconstruction error, on the structure, and on the attributes.

During training, *Suspicious* tries to minimize the reconstruction error of each auto-encoder on one of the samples ($\mathcal{V}_s$ or $\mathcal{V}_n$) while maximizing the error on the other. This leads to the definition of two loss functions, one for *Norm* and one for *Susp*:

$$L_{Norm} = \frac{\sum_{v_i \in \mathcal{V}_n} err_{Norm}(v_i)}{\sum_{v_i \in \mathcal{V}_s} err_{Norm}(v_i)},\ L_{Susp} = \frac{\sum_{v_i \in \mathcal{V}_s} err_{Susp}(v_i)}{\sum_{v_i \in \mathcal{V}_n} err_{Susp}(v_i)}. \tag{1.40}$$

Thus, both auto-encoders *Susp* and *Norm* use $\mathcal{V}_n$ and $\mathcal{V}_s$ but in a different way: nodes that are similar to the majority of the nodes in their respective sample get lower scores while nodes that are less represented in the sample get higher scores. The loss functions are minimized during training using gradient descent, as explained in the next section. As previously stated, the main difference between *Suspicious* and Dominant lies not only in its architecture. *Suspicious* is based on two auto-encoders, as shown in Figure 1.15, while Dominant has only one, but also in the calculation of the loss function. While Dominant tries to reconstruct the whole graph with minimal error, in our model, each encoder tries to minimize its own reconstruction error.

Final Score and Decision Criterion

The reconstruction errors of the nodes for each auto-encoder are normalized in $[0, 1]$, where AE is either *Susp* or *Norm*:

$$En_{AE}(v_i) = \frac{err_{AE}(v_i) - Min_{v_j \in \mathcal{V}}(err_{AE}(v_j))}{Max_{v_j \in \mathcal{V}}(err_{AE}(v_j)) - Min_{v_j \in \mathcal{V}}(err_{AE}(v_j))}. \tag{1.41}$$

The final ranking score is then:

$$Ranking_{score}(v_i) = \frac{En_{Norm}(v_i)}{En_{Susp}(v_i)}. \tag{1.42}$$

Therefore, high scores in En_{Norm} with low scores in En_{Susp} produce a high $Ranking_{score}$ corresponding to nodes classified as anomalous, which is in line with the principle behind Suspicious, described previously.

1.3 Experiments

This section details the experimental evaluation of our framework, Suspicious, and its comparison with state-of-the-art methods. It also includes an ablation study showing the interest in combining the two auto-encoders, *Norm* and *Susp*, as well as an analysis of the impact of labeling errors in the training sample, confirming the resilience of our method.

1.3.1 Datasets

The experiments are conducted on six real-world datasets:

- Cora and PubMed are popular public network datasets [65]. In these graphs, each node is a scientific publication, and the edge represents the citation of another publication. The attributes correspond to the content of the publications represented as a bag-of-words vector.

- In Photo and Computers, each node is a product, and an edge exists if two products are often purchased together. The attributes are also bag-of-words vectors.

  In Reddit and Books, described in more detail in Section 1.1.5, the label information (anomalous/normal) has been obtained through human expertise and requires no additional modification. However, as it results

Table 1.1: Characteristics of the datasets.

Dataset	#Nodes	#Edges	#Attributes	#Anomalies $\lvert\mathcal{V}_a\rvert$		Anomaly Rate $\frac{\lvert\mathcal{V}_a\rvert}{\lvert\mathcal{V}\rvert}$	
	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	d	dataR	dataS	dataR	dataS
Cora	2708	5278	1433	180	280	0.066	0.1
PubMed	19717	44338	500	4141	1971	0.208	0.1
Photo	7487	119043	745	331	748	0.043	0.1
Computers	13381	245778	767	291	1338	0.021	0.1
Books	1418	3695	21	28		0.020	
Reddit	10984	168016	64	366		0.033	

from human decisions, it can contain mislabeling. Thus, this ground truth is not fully reliable for an experimental evaluation.

The evaluation of Suspicious requires datasets including anomalous elements. We consider three types of anomalies. In addition to organic anomalies provided by humans, we follow the experimental protocols introduced by Kumagai et al. [38] and Liu et al. [45] to define, respectively, rare class anomalies and synthetic anomalies for the first four previous datasets (Cora, PubMed, Photo and Computer). Thus, two versions have been generated for each dataset *data* denoted respectively by *dataR* and *dataS*. The characteristics of our datasets are summarized in Table 1.1: number of nodes ($|\mathcal{V}|$), edges ($|\mathcal{E}|$), attributes (d), anomalies ($|\mathcal{V}_a|$) and anomaly rate ($\frac{|\mathcal{V}_a|}{|\mathcal{V}|}$).

Anomaly Types

Rare class anomalies: Each of the datasets, Cora, PubMed, Photo, and Computer, contains several classes of nodes. Following the protocol of [38], rare class anomalies can be defined by relabeling the elements of the smallest class as anomalies, while all the nodes of the other classes are considered normal. In this way, a dataset named *dataR*, suitable for binary classification with class imbalance, is created.

Synthetic anomalies: The same datasets, Cora, PubMed, Photo, and Computers, are also modified to obtain contextual and structural anomalies following the methodology proposed in [45] and fully detailed in Section 1.1.5:

- **Contextual anomalies** are created by replacing the attribute vector of a node with the most different node attribute vector out of a range of randomly selected nodes.
- **Structural anomalies** are created by adding edges to the graph to create k cliques of k nodes with $k = \sqrt{0.05 \times |V|}$.

The ratio of created anomalous nodes is 5% for each type of anomaly, resulting in a total of 10% of anomalous nodes of both types, contextual and structural, as indicated in Table 1.1 (Anomaly rate) and resulting in the dataset called *dataS*.

Introducing Labeling Errors

Each model as access to the whole graph G, but only to a limited set of labels for training $\mathcal{V}_l$, while the performances are evaluated on $\mathcal{V} \setminus \mathcal{V}_l$. In a realistic fraud detection application, the number of available labeled nodes would be low, and the labels could be erroneous. To reflect these hypotheses, we propose the following protocol:

As illustrated in Figure 1.16, the set of labeled nodes $\mathcal{V}_l$ is randomly sampled from $\mathcal{V}$. To create $\mathcal{V}_l$, $\mathcal{V}$ is sampled according to a rate R equal to 10% by taking $R \times |\mathcal{V}_a|$ anomalous nodes, where $\mathcal{V}_a \subset \mathcal{V}$ is the set of the nodes that are truly anomalous, and $R \times |\mathcal{V} \setminus \mathcal{V}_a|$ normal nodes.

As a reminder, $\mathcal{V}_l$ is split into two disjoint subsets, $\mathcal{V}_s$ and $\mathcal{V}_n$, containing nodes identified, respectively, as fraudulent and normal, when there are no labeling errors $\mathcal{V}_n = \mathcal{V}_l \setminus \mathcal{V}_a$ and $\mathcal{V}_s = \mathcal{V}_l \cap \mathcal{V}_a$. To evaluate the resilience of the methods, we then introduce errors in labels: the label of some nodes in

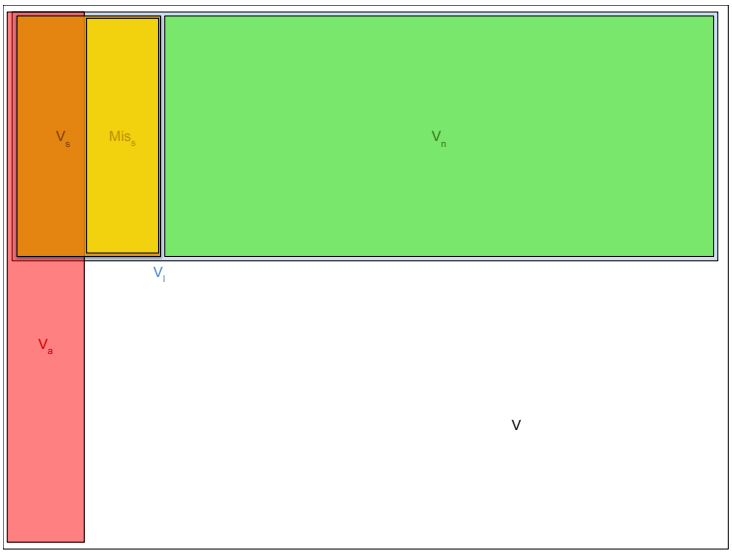


Figure 1.16: Illustration of the sets used to model labeling error. With $\mathcal{V}_a$ (red) the set anomalous nodes, the labeled set $\mathcal{V}_l$ (blue) split into two subsets, $\mathcal{V}_s$ (orange) and $\mathcal{V}_n$ (green) containing nodes labeled respectively as anomalies and normal, and Mis_s (yellow) the normal nodes mislabeled as anomalies in $\mathcal{V}_s$

$\mathcal{V}_l$ is swapped. The proportion of normal nodes wrongly labeled fraudulent is denoted Mis_s, whereas the proportion of anomalous nodes labeled normal is Mis_n:

$$Mis_n = \frac{|\mathcal{V}_a \cap \mathcal{V}_n|}{|\mathcal{V}_n|}, \text{ and } Mis_s = \frac{|\mathcal{V}_s \setminus \mathcal{V}_a|}{|\mathcal{V}_s|}. \tag{1.43}$$

Due to our task, the data is imbalanced: the number of normal nodes $|\mathcal{V}_n|$ is much larger than the number of frauds $|\mathcal{V}_a|$. Thus, Mis_n (which is upper bounded by $|\mathcal{V}_a|/|\mathcal{V}_n|$) is always small and can be neglected. Therefore, in the experiments, it is taken as equal to 0, which also decreases the number of parameters to test. The proportion Mis_s takes values varying between 0 and 0.5 in the experiments.

Finally, 10 training/test samples are produced for each dataset and each mislabeling rate Mis_s. All reported results in the following sections are averages (with standard deviation SD) over these 10 samples.

1.3.2 Settings for Suspicious and Baselines

Our methodological framework, Suspicious, is evaluated in three settings depending on the version of the auto-encoder (AE) used: ENC and DEC in Equation 1.38 can be: GCN [36], GraphSage [74] or SGC [78]. This leads to three variants, respectively denoted Ours-GCN, Ours-Sage, and Ours-SGC, in the following sections. They are compared with the following unsupervised (u) and semi-supervised (s) state-of-the-art methods:

- **Kumagai et al.** (s): detailed in Section 1.1.8, is a semi-supervised graph embedding method that uses the AUC regularizer as part of the loss to minimize the volume of a hypersphere that encompasses labeled normal nodes [38]. As Kumagai has been shown to outperform OCGNN [76], OSVM [64], Doc-N [61], Deep Walk [56], and ImVerde [79] on these datasets, we do not consider these methods as additional baselines since by outperforming Kumagai et al., Suspicious will also outperform them.

- **GNN classifiers** (s): detailed in Section 1.1.4, are three graph neural networks, GCN [36], GIN [80] and GAT [74] used as node classifiers.
- **BWGNN** (s): described in Section 1.1.8, is a graph neural network designed for anomaly detection [72].
- **Dominant** (u): detailed in Section 1.1.7, is an unsupervised method of graph reconstruction based on an auto-encoder which calculates anomaly scores as the sum of the reconstruction errors made on the attributes and the graph structure [17].
- **AnomalyDAE** (u): detailed in Section 1.1.7, is an improved version of Dominant [19]. It has been empirically demonstrated in [45] that AnomalyDAE consistently outperforms LOF [11] and IF [44]. For this reason, we do not integrate them into our evaluation.
- **Anomalous** (u): detailed in Section 1.1.7, is a method that uses CUR decomposition and residual analysis for computing a final score defined by the norm of its reconstruction residual [55].
- **CoLA** (u): detailed in Section 1.1.7, is a method that produces an embedding of the graph through a GNN and then uses contrastive learning to compute an anomaly score [47]. Moreover, as CoLA is based on the same approach as [89] and provides equivalent results, this last one is not included.

1.3.3 Evaluation Parameters and Metrics

An implementation of the method from Kumagai[2] was used with the parameters published in [38]: an embedding of dimension 4 for Books since it has fewer attributes, and 32 for the other datasets, a maximum of 500 repetitions with an early stopping mechanism, and the center C defined as the average

[2]https://github.com/tuananh0305/GCN_ANOMALY_DETECTION

of the embeddings of the nodes labeled as normal after the first layer of the model.

For Dominant, AnomalyDAE, CoLA, and Anomalous, the pygod implementation from BOND [45] was used with the publication settings, a two-layer GCN as an encoder and a one-layer GCN as an attribute decoder, an embedding dimension of 32, 100 repetitions, and $\alpha = 0.5$ for AnomalyDAE and Dominant.

For Anomalous, the gradient descent is used instead of closed-form optimization provided in the official implementation to help it perform better and faster in our task. For the GNN classifiers, the implementation benchmark [48] was used and modified to include mislabeling errors and a training set with the same size as ours.

Our framework was implemented using parts of the Dominant implementation and models from PyTorch Geometric[22]; the code and datasets are publicly available[3].

The implementations of Suspicious use the same parameters as our baselines: an embedding dimension of 4 for Books and 32 for the others, a dropout rate of 0.5 for both auto-encoders, 100 repetitions, and a learning rate of 0.005.

As in the literature, results are evaluated using AUC (Area Under the ROC curve) scores on the unlabeled nodes. ROC AUC is especially useful in anomaly detection because it avoids the need to set a fixed threshold for classification, which can be difficult to determine in imbalanced datasets where anomalies are rare. It measures how well the model distinguishes between true positives (anomalies) and false positives across all possible thresholds. AUC values range from 0.5 (random guessing) to 1 (perfect classification), with higher values indicating better anomaly detection performance.

For each dataset and each error rate $Mis_s \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$, we randomly select 10 train samples $\mathcal{V}_t$ and report the average AUC, with

[3]https://src.koda.cnrs.fr/labhc/code4publications/2023-ICTAI-suspicious

standard deviation, computed on the corresponding 10 test samples $\mathcal{V} \setminus \mathcal{V}_t$.

1.3.4 Experimental Results

The results obtained by the different methods for the datasets described in Table 1.1 are presented in Tables 1.2-1.3.

Rare Class Datasets

Table 1.2: Average AUC±SD(%) on rare class datasets.

	PhotoR	ComputersR	CoraR	PubMedR
Ours-GCN(s)	93.9 ± 0.2	**99.6**±0.1	**98.1**±0.5	95.2 ± 0.2
Ours-Sage(s)	97.1±0.9	99.5 ± 0.1	95.6 ± 0.9	95.3±0.2
Ours-SGC(s)	**97.3**±0.6	**99.6**±0.2	95.7 ± 1.4	**95.5**±0.3
BWGNN(s)	92.1 ± 4.6	72.5 ± 10.5	48.4 ± 12.4	88.8 ± 0.3
GIN(s)	68.0 ± 18.9	93.1 ± 1.2	88.3 ± 0.1	89.0 ± 0.1
GAT(s)	96.8 ± 0.2	98.4 ± 0.2	97.0 ± 0.5	91.7 ± 0.4
GCN(s)	95.6 ± 0.2	98.5 ± 0.0	98.0±0.0	92.5 ± 0.0
Kumagai(s)	94.4 ± 1.8	97.2 ± 5.5	94.5 ± 1.8	94.3 ± 0.3
AnomalyDAE(u)	51.8 ± 0.1	56.4 ± 0.1	47.9 ± 0.4	61.8 ± 0.2
Dominant(u)	51.5 ± 0.0	46.3 ± 0.0	48.7 ± 0.4	51.0 ± 0.0
Anomalous(u)	54.1 ± 6.8	49.1 ± 3.4	53.0 ± 1.2	58.9 ± 1.9
CoLA(u)	55.8 ± 3.1	66.2 ± 1.2	38.3 ± 7.3	39.2 ± 3.3

Table 1.2 shows the results obtained for the rare class datasets in the case of a perfectly labeled training sample *i.e.* when Mis_s equals 0. Our framework obtains the best results across all datasets. However, most semi-supervised methods accurately detect the anomalies and obtain relatively equivalent results to ours, while the unsupervised methods designed for synthetic anomalies completely fail to detect the rare class anomalies.

Table 1.3: Average AUC±SD(%) on synthetic and organic datasets.

	PhotoS	ComputersS	CoraS	PubmedS	Reddit	Books
Ours-GCN(s)	76.5 ± 2.3	**80.1**±4.0	78.85±3.7	**84.3**±2.6	**64.3**±1.7	58.0 ± 8.0
Ours-Sage(s)	72.3 ± 1.0	73.7 ± 0.4	75.8 ± 2.5	71.1 ± 1.0	63.5±2.8	**60.9**±5.1
Ours-SGC(s)	70.7 ± 1.8	71.8 ± 1.1	76.4 ± 2.7	69.1 ± 0.6	62.5±4.8	58.1±7.3
BWGNN(s)	55.3±8.3	64.0 ± 1.9	66.3 ± 1.2	69.5 ± 0.5	55.8 ± 2.1	54.2 ± 3.6
GIN(s)	51.2 ± 3.9	59.1 ± 3.6	32.3 ± 0.0	70.9 ± 6.4	48.9 ± 7.3	50.0 ± 0.0
GAT(s)	53.6 ± 0.8	70.3 ± 3.3	66.7 ± 0.9	72.0 ± 4.5	59.7 ± 3.7	50.0 ± 0.0
GCN(s)	57.0 ± 0.3	65.3 ± 0.3	40.0 ± 0.2	73.9 ± 0.7	62.0 ± 0.1	50.0 ± 0.0
Kumagai(s)	50.2 ± 2.3	53.2 ± 1.9	67.9 ± 2.8	57.0 ± 1.3	52.7 ± 0.7	43.4 ± 1.8
AnomalyDAE(u)	**77.5**±0.0	75.5 ± 0.0	77.5 ± 0.0	75.7 ± 0.1	48.6 ± 3.7	55.2 ± 6.7
Dominant(u)	63.6 ± 0.0	65.3 ± 0.0	**83.9**±0.0	81.2 ± 0.0	56.1 ± 0.0	38.9 ± 1.2
Anomalous(u)	48.5 ± 2.7	49.3 ± 0.08	33.6 ± 0.9	37.0 ± 0.02	53.0 ± 0.0	47.3 ± 3.1
CoLA(u)	58.5 ± 1.6	56.6 ± 0.08	67.0 ± 1.8	73.6 ± 2.3	53.0 ± 1.3	50.0 ± 0.0

Synthetic Class Datasets

The results obtained on the synthetic class datasets with perfectly labeled training samples ($Mis_s = 0$) are summarized in Table 1.3. AnomalyDAE gets the best results on PhotoS, and only our framework, Suspicious, reaches equivalent results. Dominant provides the best result on CoraS, while our framework arrives second, before AnomalyDAE, with significantly better performances than all the other methods. Finally, Ours-GCN obtains the best results on PubmedS and ComputerS.

We can observe that semi-supervised methods struggle to reach performances equivalent to those of the unsupervised methods specifically designed for these types of anomalies. However, Suspicious, which does not belong to this family of methods, still manages to achieve performances equivalent to the best-performing method, AnomalyDAE.

Organic Datasets

Finally, the results obtained on the organic datasets are also presented in Table 1.3. All of our methods obtain the best results on both Books and Reddit. While the GCN reaches equivalent performances on the Reddit dataset, all the other methods fail to find the anomalies in Books.

The poor global results on these datasets can be explained by the nature of the anomalies, as they are the result of human decisions. Indeed, the ground truth itself can contain mislabeling [87], and the labels can be inconsistent due to the diversity of the annotators.

However, since our method can identify a portion of those anomalies, it shows to be more appropriate for reproducing human reasoning than the other state-of-the-art methods, supervised or not. Nevertheless, we exclude those datasets from the complementary studies related to the impact of the parameter α and mislabeling rate Mis_s since it is not pertinent to conduct them on datasets with poor results.

1.3.5 Ablation Study

The aim of this ablation study is to check the interest of combining both auto-encoders *Norm* and *Susp* to identify the fraudulent elements. To that aim, we calculate a gain defined as the difference of performances, in terms of AUC score, obtained with Suspicious versus only one auto-encoder AE that can be *Norm* or *Susp* (Equation 1.44). To compute the performances of an AE, we calculate its AUC score using its reconstruction error err_{AE}.

$$Gain = AUC(Suspicious) - AUC(AE) \tag{1.44}$$

A positive difference corresponds to a gain and confirms the interest of using two auto-encoders instead of only one. Figure 1.17 shows these differences between the average AUC score calculated with $Norm$ (in red) and

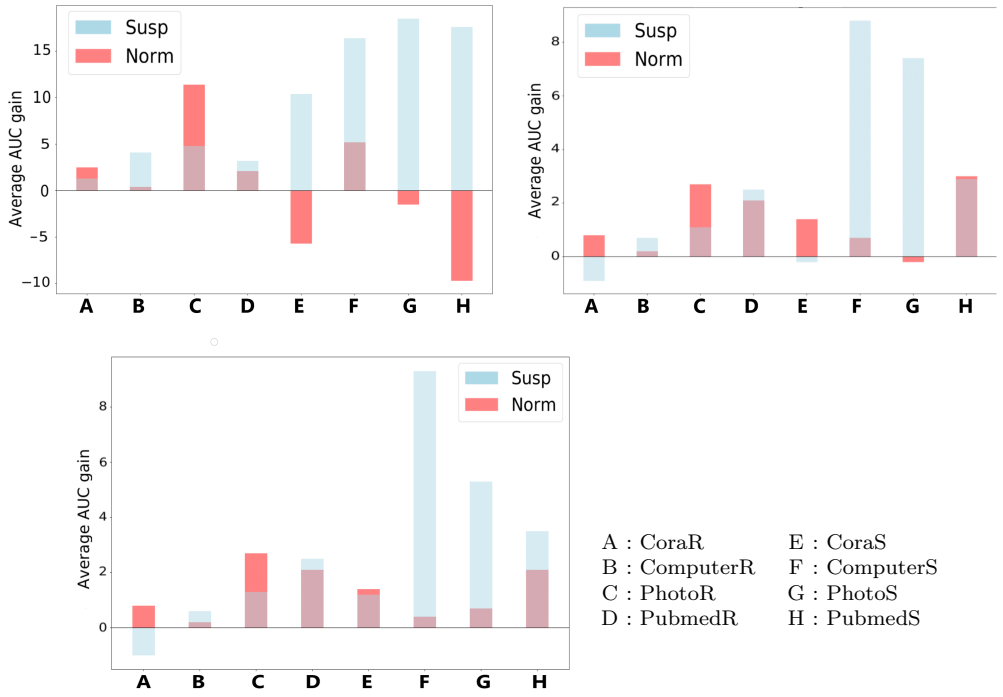


Figure 1.17: Average gain using Suspicious versus using only *Norm* (red) or *Susp* (blue) on the different datasets for the settings Ours-GCN (top left), Ours-Sage (top right) and Ours-SGC(bottom).

Susp (in blue) on the different datasets for the three settings (Ours-GCN top left, Ours-Sage top right and Ours-SGC bottom).

Ours-GCN

With Ours-GCN, the use of both auto-encoders improves the detection of rare class anomalies, comparatively to *Susp* and *Norm*, but it degrades the performances of *Norm* for synthetic anomalies on PubMedS, CoraS and, to a lesser extent, on PhotoS. This can be explained by the GCN's tendency to discard node information that is too different from its neighbors, creating a bias that makes it unable to reconstruct contextual anomalies that are part of synthetic anomalies. By incorporating two auto-encoders, this bias

can be corrected, allowing for better detection of rare class anomalies at the expense of reduced performance for contextual anomalies. It should be noted, however, that this results in an efficient and versatile model with good performance in both scenarios.

Ours-Sage and Ours-SGC

Concerning Ours-Sage and Ours-SGC, the results show that using both auto-encoders provides almost always a consistent advantage, and when it is not the case, for CoraR and PhotoS, the loss remains low while the gain can be very significant, such as for ComputerS. In conclusion, with an improvement in 42 of 48 cases (Figure 1.17), this experiment confirms the interest of combining both auto-encoders as in Suspicious. This allows all our models to gain versatility and detect both kinds of anomalies on every dataset.

1.3.6 Impact of the Mislabeling Errors

This set of experiments aims to evaluate the resilience of suspicious to mislabeling errors, which are frequent in practice due to the difficulty experts have in identifying frauds. It consists of studying the variation of the AUC score in function of Mis_s, defined in Equation 1.43. The results obtained on rare anomaly datasets are presented in Figure 1.18, while those obtained on the synthetic anomaly datasets are presented in Figure 1.19.

Since unsupervised methods cannot be affected by labeling errors, their results are included only for information. The Kumagai method has been excluded from this study due to its extremely long execution time, as well as CoLA due to its low performance. In Figure 1.18, we can observe that our models remain among the best-performing methods, no matter the value of Mis_s on rare type anomaly datasets. In Figure 1.19, we notice that, except for Ours-GCN, our framework remains stable no matter the value of Mis_s. It is important to note that Ours-GCN only seems to be affected by

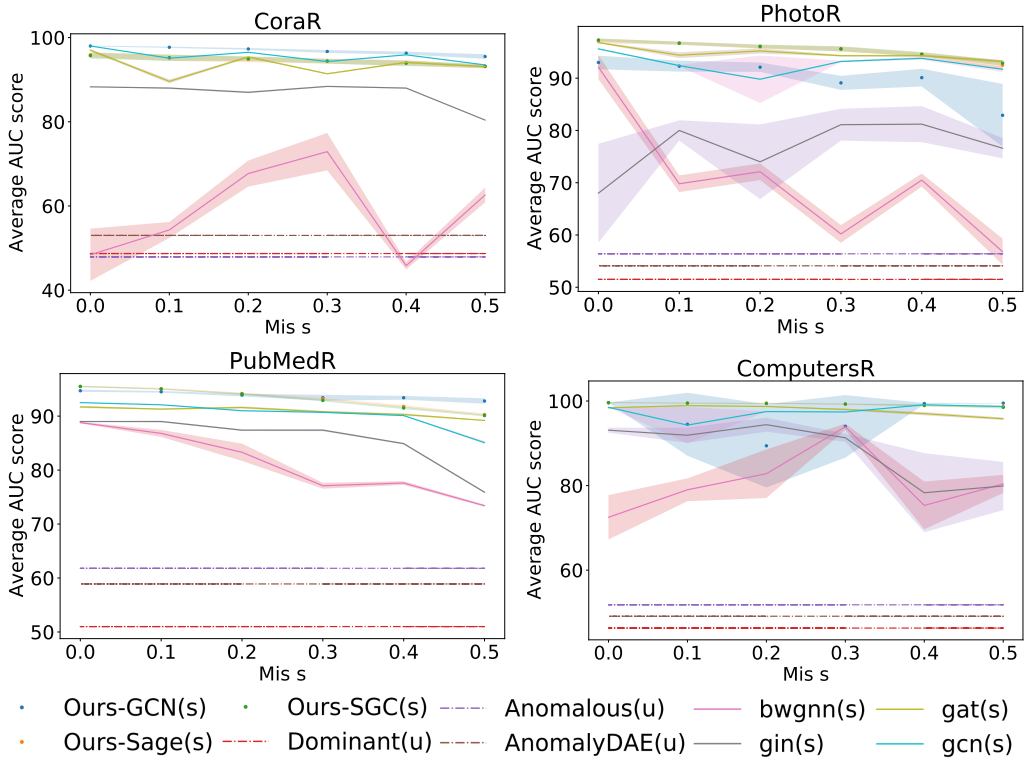


Figure 1.18: Evolution of average AUC score in function of Mis_s on rare anomaly datasets.

Mis_s when it outperforms the other models and never truly falls below their already satisfying results.

These figures show that our model is not impacted by errors occurring in the subset $\mathcal{V}_s$ of $\mathcal{V}_t$ used for its training as long as the labeling error rate remains below 40%, and even for higher rates, it outperforms most semi-supervised methods.

1.3.7 Impact of a Varying R

These experiments aim to understand the impact of R, the size of the labeled sample used for training, on the performances of our framework. Figure 1.20

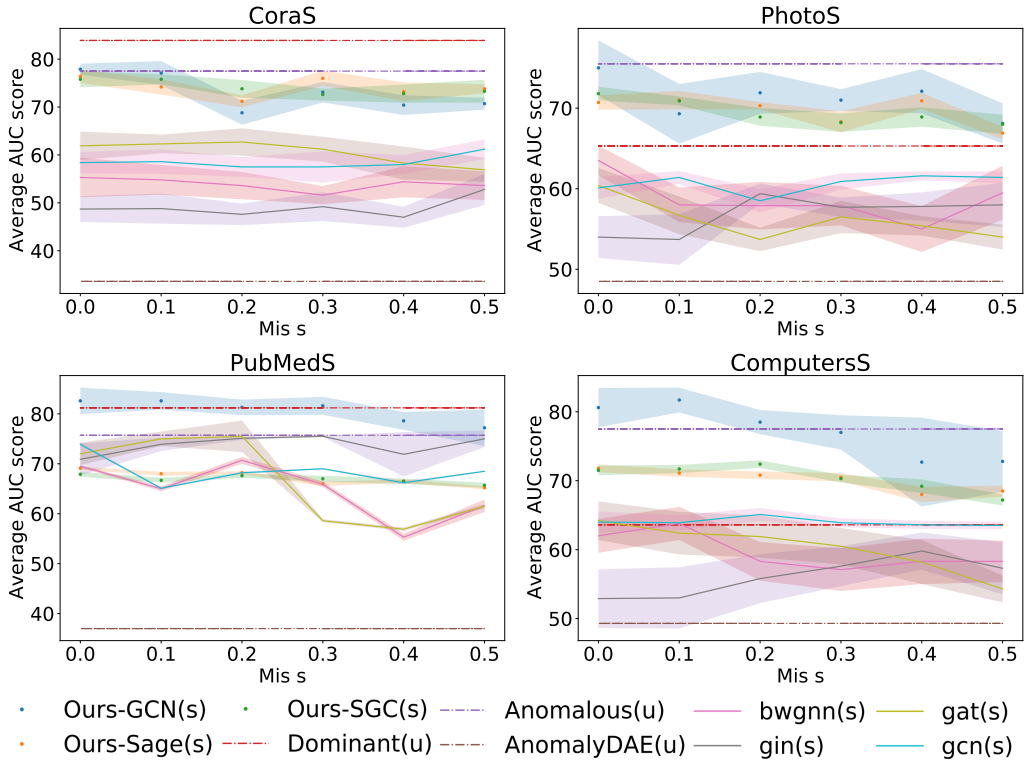


Figure 1.19: Evolution of average AUC score in function of Mis_s on synthetic anomaly datasets.

shows the variation in the average AUC-score, with standard deviation, in function of R. A lower R generally results in lower performance, but most datasets reach a competitive average AUC score with a fairly low value for R. However, a higher value is needed to stabilize the standard deviation. Thus, it seems that R mainly impacts the stability of the results, but it should be noted that a value of 3% for R is sufficient to make our framework one of the best-performing methods on all datasets.

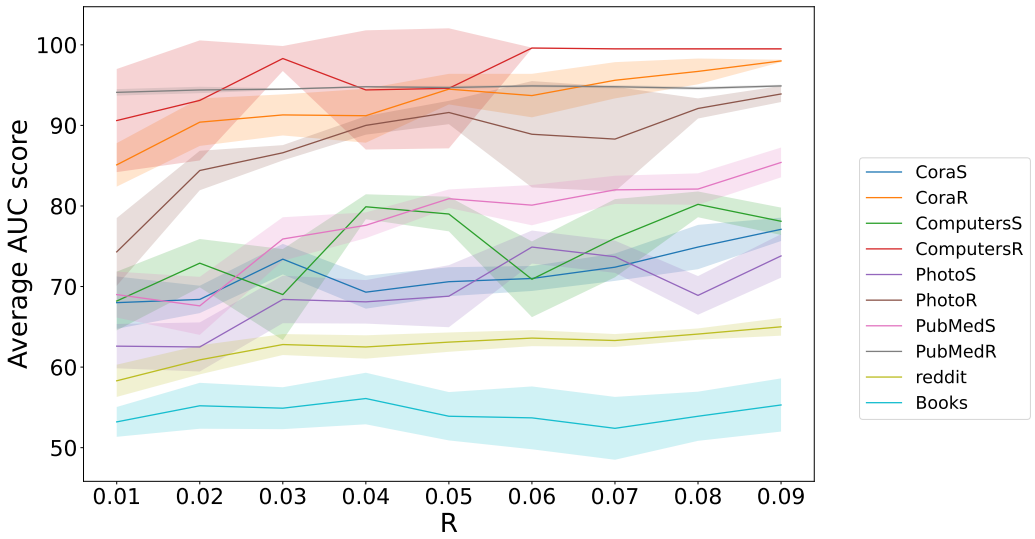


Figure 1.20: Evolution of average AUC score of Ours-GCN in function of R.

1.3.8 Impact of a Varying α

Finally, we also studied the impact of α on the results. This parameter allows the weighting of the importance given to each type of reconstruction error done either on the structure or on the attributes. Figure 1.21 presents the average AUC score of Ours-GCN for all non-organic datasets in function of α. For rare class anomalies (DataR), α has a very low impact unless it is equal to 0 or 1. For synthetic anomalies (DataS), there are lower AUC and more variations both in value and in SD (especially ComputersS and PhotoS). All in all, a value around 0.5 generally achieves good results and low SD. The stability of the results on DataR is another advantage of Suspicious over other methods, as this parameter α has an important impact on the final results of Dominant [17] and AnomalyDAE [19].

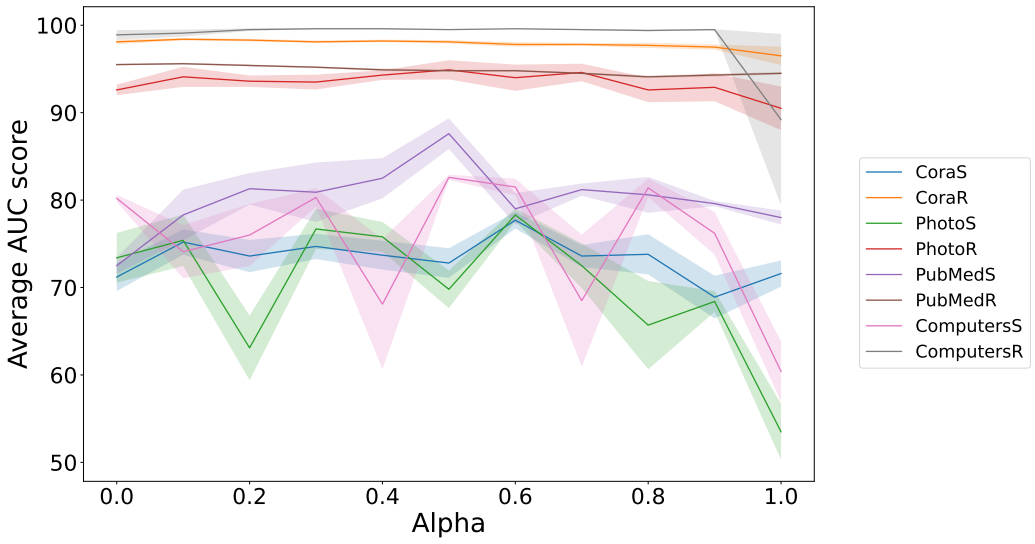


Figure 1.21: Evolution of average AUC score of Ours-GCN in function of α.

1.4 Conclusion

In this chapter, we introduced Suspicious, a novel semi-supervised framework designed for anomaly detection in attributed networks. Our work addresses critical limitations in existing methods, particularly the challenges posed by the rarity and potential mislabeling of anomalies. Unlike previous approaches that either struggle with high-dimensional graph data or require extensive labeled datasets, Suspicious leverages dual auto-encoders to enhance the detection of anomalies by focusing on both normal and fraudulent patterns.

The architecture of Suspicious distinguishes itself by employing two graph auto-encoders: Norm and Susp. The Norm auto-encoder reconstructs the graph to prioritize normal nodes, while Susp focuses on identifying and accurately reconstructing fraudulent nodes. This dual approach allows us to more effectively discriminate between normal and anomalous nodes, even in the presence of labeling errors. The final anomaly score is derived from the reconstruction errors of both auto-encoders, ensuring a robust and resilient detection mechanism that adapts to varying anomaly definitions.

Our extensive experimental evaluation demonstrated that Suspicious outperforms state-of-the-art methods across diverse datasets, including those with rare-class, synthetic, and organic anomalies. We also showed that Suspicious maintains high performance even when the labeled data is sparse or contains mislabeling, a common challenge in real-world applications. This resilience to labeling errors underscores the practicality of Suspicious in operational settings where accurate and consistent labeling can be difficult to achieve.

Moreover, our framework is flexible and can be easily adapted to various types of graph neural networks (GNNs), making it a versatile tool for anomaly detection across different domains. The ability of Suspicious to handle different types of anomalies with a unified approach while remaining robust in the face of labeling inconsistencies represents a significant advancement in the field of graph anomaly detection.

In conclusion, Suspicious offers a powerful, adaptable, and resilient solution to the complex problem of anomaly detection in attributed networks. Our contributions pave the way for future research that can further enhance the detection accuracy and applicability of graph-based anomaly detection methods, particularly in environments where data labeling is challenging or unreliable.

However, while achieving detection robust performance is crucial, it is equally important to ensure that these powerful models are explainable to human experts, particularly in high-stakes domains where trust and transparency are paramount.

Chapter 2

Explaining Graph Auto-encoders

2.1 Explainable Artificial Intelligence

Artificial intelligence (AI) has progressed significantly in the past decade, leading to widespread adoption in various fields such as healthcare, security, and finance. However, the increasing complexity of AI models presents challenges related to transparency and trust. Since most of those models are "black-box," i.e., their inner workings are not interpretable by their users. Explainable AI (XAI) has emerged as a critical field to make AI systems understandable to humans. XAI helps the users by providing insights into the decision-making processes of AI models. XAI is particularly important in high-stakes applications where understanding the reasoning behind model decisions is crucial. XAI includes a range of techniques, from simple model introspection to sophisticated methods that clarify the underlying mechanisms of complex models.

2.1.1 Explainability

Explainability is a relatively recent concept [9], as shown in Figure 2.1. This illustrates both the global increase in the number of published works using the terms interpretable, XAI, or explainable and the recent tendency to replace the older term interpretability with the newer explainability. Interpretability and explainability are often used interchangeably [62, 4], leading to confusion. But some authors insist on separating them accordingly[9]:

Interpretability: Defined as the ability to explain or provide the meaning of the model's function in terms that are comprehensible to a human.

Explainability: The notion of explanation as an interface between humans and a decision-maker, serving as both an accurate proxy of the decision-maker and comprehensible to humans.

Interpretability is a passive attribute of a model that indicates how understandable the decisions of a model are to a human observer. In contrast, explainability requires additional actions or methods to elucidate or describe the predictions done by a model.

While the importance of explainability in AI is clear, the challenge lies in developing reliable methods to uncover these explanations, especially in complex models like Graph Neural Networks (GNNs). In the next section, we explore different approaches to explainability in machine learning, focusing on how explanations can be derived from various model architectures.

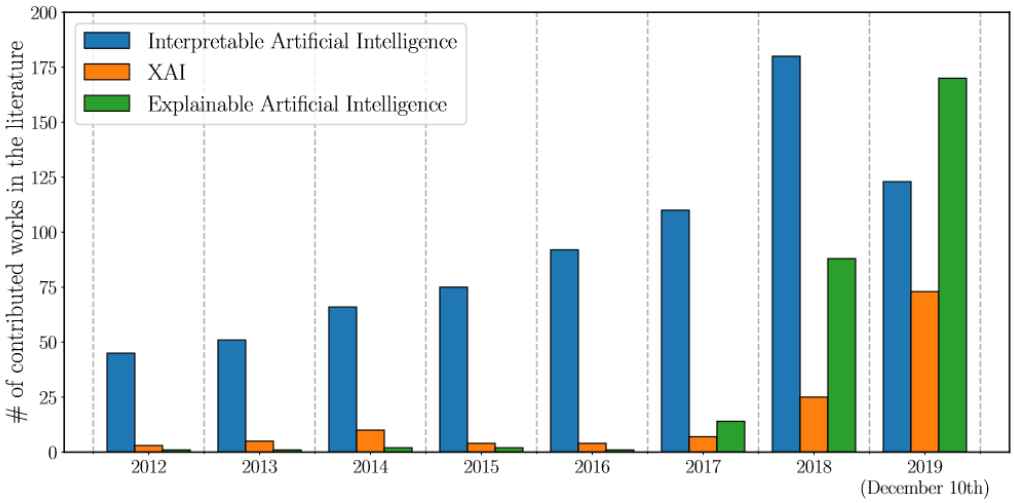


Figure 2.1: Evolution of the number of total publications whose title, abstract, and/or keywords refer to the field of XAI in recent years. Figure from [9].

2.1.2 Explaining Machine Learning Models

Explanations Taxonomy

The various interpretations of what an explanation is and how it is obtained have led to a great variety of models to generate them. The general taxonomy proposed by [52] and followed by [53, 5, 1] is defined as:

- **Intrinsic or post hoc:** Intrinsic explainability refers to machine learning models that are inherently interpretable, such as short decision trees or sparse linear models. Post hoc explainability involves applying explanation methods after the model has been trained.

- **Model-specific or model-agnostic:** Model-specific explanation tools are tailored to specific types of models. For instance, tools that work exclusively with neural networks are model-specific. Model-agnostic tools can be applied to any machine-learning model and are used after

the model has been trained.

- **Local or global:** Local explanation methods focus on elucidating individual predictions, whereas global explanations aim to clarify the overall behavior of the model.

When explaining a GNN, it is important to add the elements of the graph that are used as explanations to our taxonomy:

- **Element of explanation:** Graph explainers vary in which elements of the graph they identify as part of the explanation; this can include any combination of nodes $\mathcal{V}$, edges $\mathcal{E}$, and features $\mathbf{X}$.

Explainer	**Model-specific/ agnostic**	**Element of explanations**	**Method**
Integrated Gradients [69]	Model-specific	$\mathcal{V}/\mathbf{X}$	Gradient
Grad [70]	Model-specific	$\mathcal{V}/\mathbf{X}$	Gradient
GuidedBP [7]	Model-specific	$\mathcal{V}/\mathbf{X}$	Gradient
GNNExplainer [83]	Model-agnostic	$\mathcal{V}/\mathcal{E}/\mathbf{X}$	Perturbation
SubGraphX [85]	Model-agnostic	$\mathcal{V}/\mathcal{E}$	Perturbation
GraphLIME [33]	Model-agnostic	$\mathbf{X}$	Surrogate
PGM-Explainer [75]	Model-agnostic	$\mathcal{V}$	Surrogate

Table 2.1: Characteristics of some explainers.

Following the established general framework for understanding explainability, we now turn our attention to Graph Neural Networks (GNNs). GNNs present unique challenges and opportunities for explainability due to their ability to model graph-structured data, where both the structure and attributes play crucial roles in predictions. In the next section, we focus on explainability methods tailored to GNNs, particularly those used in node and graph classification tasks.

2.1.3 Explaining GNN

Explainability in Graph Neural Networks (GNNs) is significantly constrained by the availability of datasets with explainable ground truth [84, 5, 1]. Consequently, the limited existing works focus either on the node or graph classification tasks, with no substantial focus on the anomaly detection task. We will concentrate on the node classification task as it is the nearest task available with already existing literature.

Definitions and Problem Formalization

Let $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ be an attributed network defined by the set of nodes $\mathcal{V} = \{v_1, \ldots, v_n\}$, the set of edges $\mathcal{E}$ represented by a symmetric adjacency matrix $\mathbf{A} = (a_{i,j}) \in \{0,1\}^{n \times n}$ where $a_{i,j} = 1$ if there is an edge between the nodes i and j and $a_{i,j} = 0$ otherwise, and the attribute matrix $\mathbf{X} = (x_{i,j}) \in \mathbb{R}^{n \times d}$ which ith row $\mathbf{x}_i$ represents the attribute vector of v_i. In the same way, $\mathbf{a}_i$ denotes the ith row of $\mathbf{A}$. In the following, bold upper case letters denote matrices, and bold lower case letters denote vectors.

Given a model f which produces a prediction $\hat{y} = f(v_i, G)$ for a node v_i, in the more general case, an explanation of this prediction is a vector $\mathrm{Imp}(f, v_i)$ which contains an importance weight for each node, edge, and attribute of the graph. Higher weights are associated with the most important elements for the prediction $f(v_i, G)$.

However, these importance weights for $f(v_i, G)$ are generally limited to elements (edges, nodes, and their attributes) located in the h-hop neighborhood subgraph of the node v_i. The h-hop neighborhood subgraph is the induced subgraph of G by all the nodes that are at a distance less or equal to h from v_i. The notion of distance in a graph is based on paths between nodes; by only traveling through edges, how many edges must be crossed to

reach the destination nodes. It is denoted as:

$$\mathrm{SUB}(v_i, h) = (\mathcal{V}_{\mathrm{SUB}}(v_i, h), \mathcal{E}_{\mathrm{SUB}}(v_i, h), X_{\mathrm{SUB}}(v_i, h)). \tag{2.1}$$

Also, we split the importance weights into three vectors: $\mathrm{Imp}_{\mathcal{V}}(f, v_i)$ is the importance vector with one weight for each node of $\mathrm{SUB}(v_i, h)$, $\mathrm{Imp}_{\mathcal{E}}(f, v_i)$ the weights for the edges of $\mathrm{SUB}(v_i, h)$ and $\mathrm{Imp}_{\mathbf{X}}(f, v_i)$ the weights for the attributes of node v_i. In the following, we will drop h and f from the notations for simplicity.

Most GNN explainers are model-agnostic and post-hoc models adapted from image-based tasks. These can be categorized into three primary groups: gradient-based, perturbation-based, and surrogate-based.

2.1.4 Gradient Based Explainers

Employing gradients [69, 70, 7, 23] to explain deep models is the most straightforward solution as it directly uses the learned weights of a neural network to provide explanations for its predictions. This approach is widely used in image and text tasks. The key idea is to use the gradients of the output with respect to the input features as the approximations of input importance. Intuitively, the gradient measures how much a small change in the input affects the output, highlighting which parts of the input have the most significant impact on the model's predictions, with larger gradients indicating higher importance. Since these methods are simple and general, they can be easily extended to the graph domain by only considering the feature matrix.

All explanations of these methods are limited to feature explanation and node explanation when we aggregate the weight of a node's features but do not take into account the structure of the graph.

Explainer	Model-specific/ agnostic	Element of explanations	Method
Integrated Gradients [69]	Model-specific	$\mathcal{V}/\mathbf{X}$	Gradient
Grad [70]	Model-specific	$\mathcal{V}/\mathbf{X}$	Gradient
GuidedBP [7]	Model-specific	$\mathcal{V}/\mathbf{X}$	Gradient

Table 2.2: Characteristics of gradient-based explainers. Extract from Table 2.1

Grad

Grad [70] is one such method; it measures node importance as the square values of gradients directly through back-propagation, assuming that higher absolute gradients indicate more important corresponding input features.

Integrated Gradients

Integrated gradients (IGrad) [69], initially designed for images, uses a fully black image as a neutral state to initialize the gradient. It then measures how predictions change when the image is interpolated with the image being explained and generates importance scores by accumulating the gradient effect from the baseline to the actual features. For the graph adaptation of this method, the baseline graph is a copy of the graph being explained, with all features replaced by the value 1. Then IGrad measures how predictions change with feature changes and generates importance scores by accumulating the gradient effect along a path from the baseline to the actual features.

Guided Back Propagation

Guided Back Propagation (GuidedBP) [7] employs the values of positive gradients as the importance scores of different input features. Positive gradients indicate how much increasing the value of the feature would contribute to the output, aligning with the desired prediction. GuidedBP aims to provide a clearer picture of the features that influence the model's decision-making

by eliminating negative influences.

Gradient-Based Explainer Evaluation

The few works comparing those methods to other state-of-the-art methods [5, 1] reach the same conclusion: those methods are the most efficient. Even though they may have lower explanation performances compared to the best methods described below, they require much less time and memory resources. In general, these three gradient-based methods are relatively equivalent.

2.1.5 Perturbation-Based Methods

Perturbation-based methods [85, 83] often explain deep image models by studying output variations with different input perturbations. Image-based perturbation methods learn a generator to create a mask to select necessary input pixels to explain deep image models. However, such methods cannot be directly applied to graph models. Contrary to gradient-based methods that took already functioning models from other domains, perturbation-based methods required new dedicated methods. For GNNs, as shown in Figure 2.2, most methods use the already trained GNN and give it masked information (perturbation) to see how it affects the model prediction. Masking important information should impact the prediction, while unimportant ones should leave it unaffected.

Explainer	**Model-specific/ agnostic**	**Elements of explanations**	**Method**
GNNExplainer [83]	Model-agnostic	$\mathcal{V}/\mathcal{E}/\mathbf{X}$	Perturbation
SubGraphX [85]	Model-agnostic	$\mathcal{V}/\mathcal{E}$	Perturbation

Table 2.3: Characteristics of perturbation-based explainers. Extract from Table 2.1

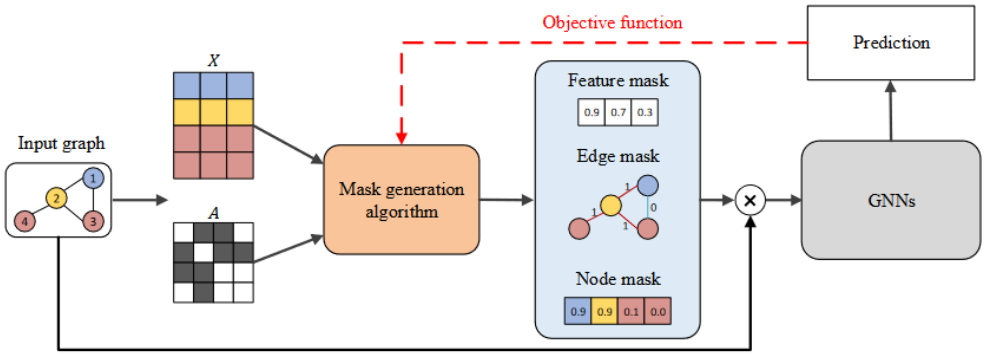


Figure 2.2: The general pipeline of perturbation-based methods. Figure from [84]

SubGraphX

SubGraphX [85] seeks to identify important subgraphs by measuring the impact on prediction when the subgraph is deleted. This creates explanation masks for both edges and nodes in deep graph models. It uses the Monte Carlo Tree Search (MCTS) algorithm [68] to explore and select the subgraph, a heuristic search method from game theory. It randomly explores the graph, adding nodes, testing the usefulness of each node, and pruning the unhelpful ones, as shown at the bottom of Figure 2.3.

This method relies heavily on the score function used to evaluate the nodes. SubGraphX uses Shapley values for this purpose.

Shapley values:

Shapley values originate from cooperative game theory [66]. In a game that requires the cooperation of several players to win, Shapley values offer a fair approach to distributing gains among players based on their contributions to the overall success of the coalition. In the realm of explainable AI, the features of a model are viewed as players in a cooperative game, with the prediction serving as the total payoff. These values are the only way to

guarantee all the following properties at the same time: efficiency (ensuring full distribution of the total gain), symmetry (granting equal value to equal contributions), dummy player (assigning zero value to non-contributing players), and additivity (ensuring that the value in combined games is the sum of values in individual games).

In the context of graph model explanation tasks, we can consider the prediction of a GNN as the game gain and the various graph structures as the players. By identifying the players that make the greatest contributions to the gain, we can determine the most important subgraph for the prediction.

For a graph G with n nodes and a trained GNN f($\cdot$), the Shapley value for a subgraph G_i with k nodes is computed as:

$$\phi(G_i) = \sum_{S \subseteq P \setminus \{G_i\}} \frac{|S|!(|P| - |S| - 1)!}{|P|!} m(S, G_i), \tag{2.2}$$

$$m(S, G_i) = f(S \cup \{G_i\}) - f(S), \tag{2.3}$$

where S is a possible coalition set of players, P is the set of players, and $m(S, G_i)$ represents the contribution of G_i given S.

Computing Shapley values directly is time-consuming because it requires evaluating every possible coalition of nodes and edges in the graph. To address this, SubGraphX approximates Shapley values using the GNN architecture. Only nodes within L-hops are considered for aggregation. Monte Carlo sampling is used to compute $\phi(G_i)$. For each sampling step i, a coalition set S_i is sampled from $P \setminus \{G_i\}$, and its contribution $m(S_i, G_i)$ is calculated. $\phi(G_i)$ is obtained using the average contribution score over multiple samples:

$$\phi(G_i) = \frac{1}{T} \sum_{t=1}^{T} (f(S_i \cup \{G_i\}) - f(S_i)), \tag{2.4}$$

Where T is the total number of sampling steps. Nodes not in the coalition

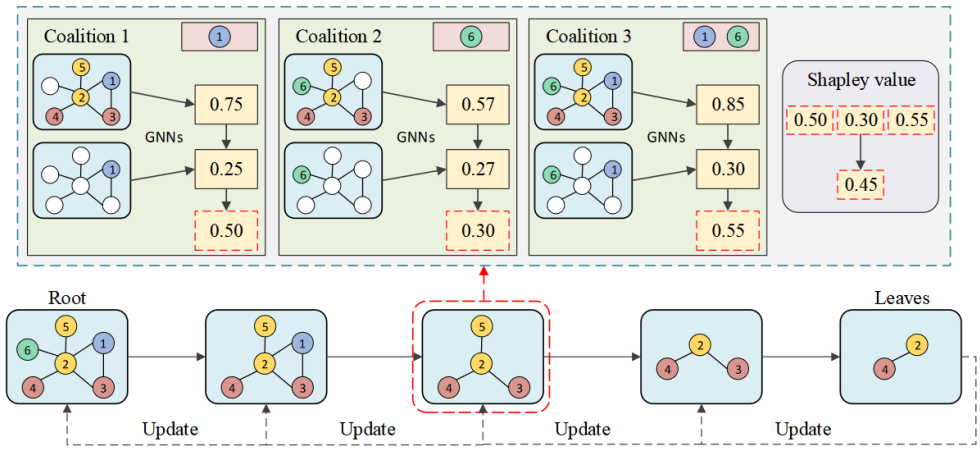


Figure 2.3: An illustration of SubGraphX [85]

or subgraph have their features set to zero, and the graph is fed to the GNN to get the predicted probability. This process is repeated to approximate the Shapley value efficiently, as shown in the top of Figure 2.3.

GNNExplainer

GNNExplainer (GNNEx) [83] is designed to identify the important elements of a graph, nodes, edges, and attributes by measuring the impact of deleting these elements on the predictions made by the original GNN. Unlike SubGraphX, which tests every possible combination, GNNExplainer trains a model to predict which elements are important. To do so, it functions in three steps:

Subgraph Identification: The selection of the subgraph is based on optimizing a mutual information criterion to ensure the subgraph contains the most informative elements.

For a given node v_i, GNNExplainer aims to find a subgraph $G_S \subseteq \text{SUB}(v_i, l)$, where l is the number of layers in the target GNN, along with the associated features $\mathbf{X}_S = \{x_j \mid v_j \in G_S\}$ that are crucial for the GNN's pre-

diction $\hat{y}_i$. The authors assume that $\mathbf{X}_S$ is a small subset of $\mathbf{X}$. Importance is formalized using mutual information (MI), and the GNNExplainer model is formulated accordingly:

$$\max_{G_S} \mathrm{MI}(\hat{y}_i, (G_S, \mathbf{X}_S)) = H(\hat{y}_i) - H(\hat{y}_i \mid G = G_S, \mathbf{X} = \mathbf{X}_S) \tag{2.5}$$

Here, H represents entropy, a measure of uncertainty or randomness in a random variable, and $\hat{y}_i$ represents the prediction of the GNN for the node v_i. Specifically, $\hat{y}_i$ denotes the class label predicted by the GNN for the node v_i. For each node v_i, the Mutual Information (MI) measures how much the likelihood of predicting $\hat{y} = f(\mathrm{SUB}(v_i, l), \mathbf{X}_{\mathrm{SUB}(v_i,l)})$ changes when the computation graph of v_i is restricted to the explanation subgraph G_S and its node features are restricted to $\mathbf{X}_S$.

The explanation for prediction $\hat{y}_i$ is a subgraph G_S that minimizes the uncertainty of $f(v_i, G_S)$ when the Graph Neural Network (GNN) computation is limited to G_S. To create a concise explanation, the authors constrain the size of G_S to have at most K_M nodes, denoted as $\|G_S\| \leq K_M$. GNNExplainer aims to filter $\mathrm{SUB}(v_i, l)$ by selecting K_M edges that have the highest mutual information with the prediction.

Feature Masking: GNNExplainer determines the importance of each feature within the nodes of this subgraph. To determine which node features are most important for predicting $\hat{y}_i$, GNNExplainer trains a feature selector F for nodes in the explanation G_S. Instead of including all node features in $\mathbf{X}_S$ (i.e., $\mathbf{X}_S = \{x_j \mid v_j \in G_S\}$), GNNExplainer defines $\mathbf{X}_S^F$ as a subset of features of nodes in G_S. This subset is determined by a binary feature selector $F \in \{0, 1\}^d$:

$$\mathbf{X}_S^F = \{x_j^F \mid v_j \in G_S\}, \quad x_j^F = [x_{j,t_1}, \ldots, x_{j,t_k}] \text{ for each } F_{t_i} = 1 \tag{2.6}$$

Optimization: The explanations $(G_S, \mathbf{X}_S^F)$ are then jointly optimized to maximize the mutual information objective:

$$\max_{G_S, F} \mathrm{MI}(\hat{y}_i, (G_S, \mathbf{X}_S^F)) = H(\hat{y}_i) - H(\hat{y}_i \mid G = G_S, \mathbf{X} = \mathbf{X}_S^F) \tag{2.7}$$

This optimization ensures that the selected subgraph G_S and feature subset $\mathbf{X}_S^F$ retain the most significant information for the prediction, providing a compact and informative explanation. The method's ability to focus on a small, informative subgraph and relevant features makes GNNExplainer a powerful tool for interpreting the decisions of Graph Neural Networks.

Perturbation-based methods evaluation

When comparing performances, previous works [5, 1] found that SubGraphX produced the best results compared to all other methods. However, it also requires the longest execution time and does not consider the feature matrix. GNN explainer produced the second-best overall result, requiring a considerable amount of time but still less than SubGraphX. It covers every element of a graph and is currently considered the best method in the state-of-the-art for attributed graphs.

2.1.6 Surrogate Based Methods

Surrogate methods are utilized to approximate the model and explain it with a simpler, more interpretable model. The explanation from the simpler model is considered an approximation of the explanation of the original model.

Explainer	**Model-specific/ agnostic**	**Element of explanations**	**Method**
PGM-Explainer [75]	Model-agnostic	$\mathcal{V}$	Surrogate

Table 2.4: Characteristics of PGM-Explainer. Extract from Table 2.1

PGM-Explainer

PGM-Explainer (Probabilistic Graphical Model Explainer) [75] is a perturbation-based method designed to provide interpretable node explanations for the predictions made by Graph Neural Networks (GNNs). As shown in Figure 2.4, PGM-Explainer generates perturbed graphs and records GNN's predictions on those graphs in the data generation step. The variable selection step eliminates unimportant explained features in this data and forwards the filtered data. Those steps are similar to the methods detailed in the perturbation-based methods. Finally, a Bayesian Network, a type of interpretable Probabilistic Graphical Model (PGM), is generated to model the dependencies between the input features and the model's output.

Bayesian Network Construction: PGM-Explainer constructs a Bayesian Network to capture the relationships between node features, edges, and the GNN's predictions. This model is used to understand how different elements of the input graph influence the output. The Bayesian Network is interpretable because it represents the conditional dependencies between variables in a clear and structured manner, allowing users to see how changes in one variable might affect others.

For a given node v_i, PGM-Explainer aims to construct a Bayesian Network $\mathcal{P}$ that models the joint distribution of the nodes $\mathcal{V}$ and the GNN's prediction $\hat{y}_i$. The objective is to find the most informative nodes that explain the prediction. The dependencies are formalized using conditional probabilities.

Node Importance: PGM-Explainer determines the importance of each node by analyzing their conditional probabilities given the prediction $\hat{y}_i$. This is done by calculating the influence of each node on the prediction using the constructed Bayesian Network.

Optimization and Inference: The explanation is generated by identifying the most significant nodes that maximize the likelihood of the GNN's prediction. The process involves optimizing the conditional probability distri-

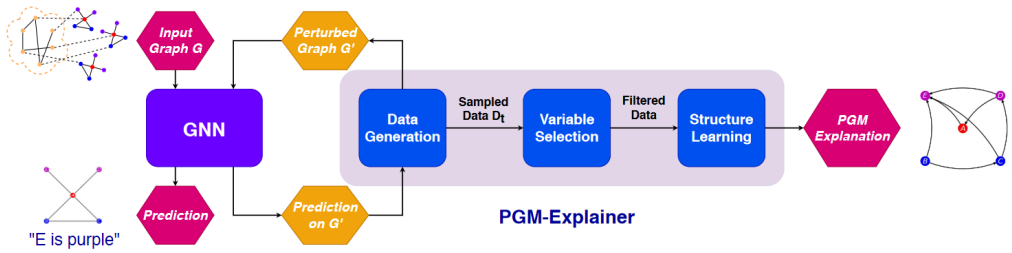


Figure 2.4: The architecture of PGM-Explainer. Figure from [75]

bution to ensure that the selected nodes provide a high-fidelity explanation.

The core optimization problem is formulated as follows:

$$\max_{\mathcal{V}_S} P(\hat{y}_i \mid \mathcal{V}_S) = \max_{\mathcal{V}_S} \prod_j P(\hat{y}_i \mid v_j \in \mathcal{V}_S) \tag{2.8}$$

Here, $P(\hat{y}_i \mid \mathcal{V}_S)$ represents the conditional probability of the prediction given the subgraph $\mathcal{V}_S$. The goal is to maximize this probability to ensure that the selected nodes provide the best explanation for the prediction.

Subgraph Selection: The final step involves selecting the subgraph $\mathcal{V}_S$ that has the highest impact on the prediction. This is done by evaluating the influence of different combinations of nodes using the constructed Bayesian Network.

$$\mathcal{V}_S = \arg\max_{\mathcal{V}'} P(\hat{y}_i \mid \mathcal{V}') \tag{2.9}$$

Surrogate evaluation

When this method is compared to the rest of the state-of-the-art [5, 1], it reached lower or equivalent performances to gradient-based methods while still taking around as much time and resources as GNN-explainer and only explaining one type of elements.

2.1.7 Measuring Explainability

Measuring explainability involves evaluating how well explanation techniques can identify the most important input features influencing the model's decisions. This section explores various metrics used to assess the quality of explanations. Among these metrics are fidelity, which examines the impact of removing important features on model performance, and precision and recall, which are used to compare explanations against known ground truths in synthetic datasets. Each metric offers a unique perspective on the explainability of GNNs, contributing to a comprehensive evaluation framework.

Fidelity

Explanations should identify input features important in the decision making process of the model. The necessity and sufficiency metrics, often referred to as fidelity+ and fidelity- respectively, were recently proposed to evaluate explanations [42] and then later adapted to the graph domain [84, 5]. Necessity refers to the difference in model performance, prediction accuracy, or probability prediction when important input features identified by explanation techniques are removed. If these features are truly influential, the model's predictions should change significantly after their removal [42, 83, 85].

Given a model f, that outputs a probability class vector of size j, $\hat{y}_i$ denotes $argmax_j(f(v_i, G))$ the class prediction of f for a node v_i, i.e., the predicted label of v_i calculated from graph data $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$. Given an explanation mask $\text{Exp}(v_i)$ derived from the importance vector $\text{Imp}(v_i)$, $G \setminus \text{Exp}(v_i)$ denotes the graph G where all the edges and attributes corresponding to a 1 in $\text{Exp}(v_i)$ are hidden (replaced by 0 in the feature and adjacency matrix). We also denote $G \cap \text{Exp}(v_i)$ as the graph where the only available information is the edges and attributes corresponding to a 1 in $\text{Exp}(v_i)$ (all other edges and attributes are replaced by 0 in the feature and adjacency matrix).

Then the necessity, more commonly referred to as fidelity+ and denoted as $fid+$, measures the change in the prediction, when the only information available to f is $G \setminus \text{Exp}(v_i)$, and sufficiency, referred to as fidelity- and denoted as $fid-$, measures the change when the only information available to f is $G \cap \text{Exp}(v_i)$:

$$fid+_{acc}(v_i) = \mathbb{1}(\hat{y}_i = y_i) - \mathbb{1}(\hat{y}_i = \hat{y}_i^{\setminus \text{Exp}}) \text{ with } \hat{y}_i^{\setminus \text{Exp}} = \text{argmax}_j(f(v_i, G \setminus \text{Exp}(v_i))), \tag{2.10}$$

$$fid-_{acc}(v_i) = \mathbb{1}(\hat{y}_i = y_i) - \mathbb{1}(\hat{y}_i = \hat{y}_i^{\text{Exp}}) \text{ with } \hat{y}_i^{\text{Exp}} = \text{argmax}_j(f(v_i, G \cap \text{Exp}(v_i))). \tag{2.11}$$

Where $\mathbb{1}$ is an indicator function that returns 1 if both predictions match, and 0 otherwise. $fid+_{\text{acc}}$ tracks the change in predicted accuracy for the original class label y_i, with higher values indicating better explanation results. Lower values for fid-$_{acc}$ indicate that less important information is removed, suggesting better explanation results, for clarity, sufficiency is often reported as $1-fid-$ so that higher scores indicate better explanation for both metrics.

To function properly, these metrics only need to evaluate the change in an expression of the performance of the model. Depending on the output of the model, fidelity can also be evaluated based on the class probability for regression tasks as they are only based on the predicted probabilities [5]:

$$fid+_{prob}(v_i) = (P_{\hat{y}_i} - P_{\hat{y}_i^{\setminus \text{Exp}}}) \text{ with } P_{\hat{y}_i^{\setminus \text{Exp}}} = f(v_i, G \setminus \text{Exp}(v_i))[y_i], \tag{2.12}$$

$$fid-_{prob}(v_i) = (P_{\hat{y}_i} - P_{\hat{y}_i^{\text{Exp}}}) \text{ with } P_{\hat{y}_i^{\text{Exp}}} = f(v_i, G \cap \text{Exp}(v_i))[y_i]. \tag{2.13}$$

Similarly, $fid+_{\text{prob}}$ tracks the change in predicted probabilities for the original class label y_i, where higher values indicate better explanation results. Lower values for $fid-_{\text{prob}}$ suggest that less critical information is removed, and as with $fid-_{\text{acc}}$, it's common to report $1 - fid-_{\text{prob}}$ so that higher scores reflect better explanation quality for both metrics.

In both cases, fidelity fundamentally measures the same concept: the impact of explanatory features on model performance.

$fid+_{\text{acc}}$ and $fid-_{\text{acc}}$ assess the change in prediction accuracy when explanatory information is removed or focused upon while $fid+_{\text{prob}}$ and $fid-_{\text{prob}}$ measure the change in class probabilities. These metrics are adapted to the specific way a model's performance is measured, whether through accuracy for classification tasks or probabilities for regression tasks. Despite these variations, the underlying goal remains consistent: tracking how the model's performance shifts in response to the provided explanations.

Precision and Recall

These metrics are proposed for synthetic datasets [82]. In synthetic datasets, although it is unknown whether the GNNs make predictions in the expected way, the rules for constructing these datasets, such as graph motifs, can serve as reasonable approximations for ground truth explanations. For any input graph, we can compare explanations with these ground truth explanations, which represent the reasons why we attribute the classification ground truth and what the model should detect to fulfill the task properly. Common metrics for such comparisons include precision, recall, general accuracy, F1 score, and ROC-AUC score. Higher values for these metrics indicate that the explanations are closer to the ground truths and can be considered better results, but these metrics cannot be applied to real-world datasets due to the lack of ground truth explanations.

To create datasets with an explanation ground truth, dedicated graph

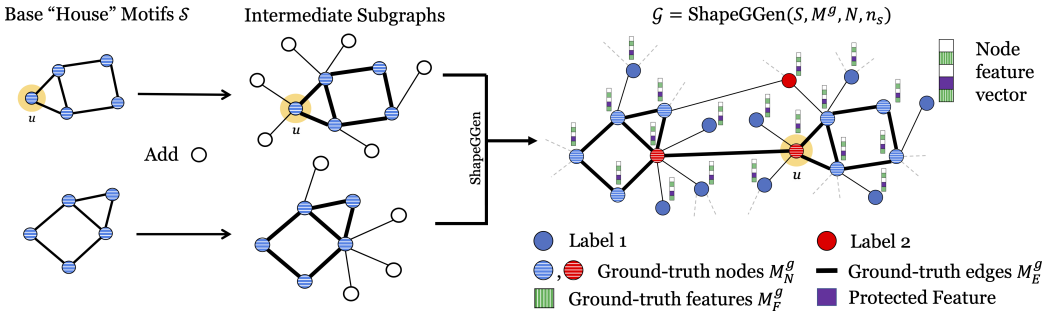


Figure 2.5: Overview of ShapeGGen graph dataset generation from [1]

generator were introduced in [83, 1].

2.1.8 Generating Explanation Ground Truth

ShapeGGen

ShapeGGen is a graph generator proposed in [1], that we will use in our experiments to create an attributed graph with a ground truth explanation that allows us to measure the performances of explainers. ShapeGGen starts by creating a set of small subgraphs (motifs) $M = \{M_1, ..., M_N\}$ with a specific structure. These subgraphs are then connected using a preferential attachment algorithm, adding nodes that connect to existing ones. Each node in the generated graph is labeled based on the number of motifs in its 1-hop neighborhood: label 1 if there is 0 or 1 motif, and label 2 if there are more. A latent variable model [29] is then used to create two kinds of node features for each node: informative features correlated to the node's label and random non-informative features independent of the label.

In addition to providing the ground truth labels for classification, ShapeGGen generates ground truth explanations for all graph nodes. This information is represented by two binary masks $\text{Gt}_{\mathcal{E}}(v_i)$ and $\text{Gt}_{\mathbf{X}}(v_i)$ for each anomalous node v_i. These are the ground truth vectors corresponding to explanation vectors $\text{Exp}_{\mathcal{E}}(v_i)$ and $\text{Exp}_{\mathbf{X}}(v_i)$.

Let the motifs within $\text{SUB}(v_i, 1)$, the 1-hop neighborhood of v_i be:

$$M_{v_i} = (\mathcal{V}_{M_{v_i}}, \mathcal{E}_{M_{v_i}}, \mathcal{X}_{M_{v_i}}) = M \cap \text{SUB}(v_i, 1). \tag{2.14}$$

Using this notation, the authors define ground truth explanation masks:

- Node explanation mask $\text{Gt}_V(v_i)$: nodes in $\text{SUB}(V_i)$ are labeled as 1 if they belong to M_{v_i} and 0 if they don't.
- Feature explanation mask $\text{Gt}_X(v_i)$: each feature in $\mathbf{x}_i$ is labeled 1 if it represents an informative[1] feature, 0 if it does not, producing $\text{Gt}_X(v_i) \in \{0, 1\}^d$;
- Edge explanation mask $\text{Gt}_\mathcal{E}(v_i)$: each edge $e = (v_j, v_k) \in \mathcal{E}_{\text{SUB}}(v_i)$ is labeled 1 if both v_j and v_k belong to $(\mathcal{V}_{M_{v_i}} \cup \{v_i\})$ and 0 otherwise, producing $\text{Gt}_\mathcal{E}(v_i) \in \{0, 1\}^{|\mathcal{E}_{\text{SUB}}(v_i)|}$.

BA-Shapes

BA-Shapes is a synthetic graph data generator used for node classification tasks, introduced in [83]. As shown in Figure 2.6, the process begins with creating a Barabasi-Albert (BA) [3] graph consisting of n nodes. A BA graph is a type of scale-free network where nodes are added sequentially, and new nodes are preferentially connected to existing nodes with higher degrees, resulting in a network with a power-law degree distribution. This BA graph serves as the base structure. To this base graph, we attach several five-node "house"-shaped motifs. Specifically, K house motifs are randomly attached to nodes in the base graph. Finally, random edges are added to the graph to add variability. In the resulting graph, nodes are classified into two categories: nodes that are part of a house motif are labeled as class 1, while nodes that are not part of any house motif are labeled as class 0.

[1] I.e., the classification label is perfectly dependent on the feature.

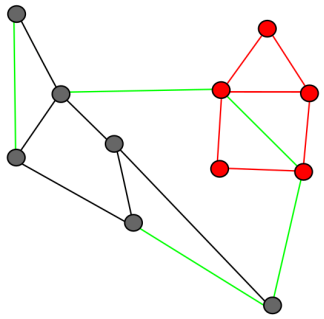


Figure 2.6: Overview of BA-Shapes graph dataset generation. it starts by generating the graph structure in black, then it add the motif in red, and finally adds random edges in green.

Real World Dataset

In [5], the authors propose that since the fidelity metric does not need any ground truth to function, it can be used to integrate real-world datasets compatible with the task being studied.

2.2 Explaining Auto-encoders

While many explainers for GNNs focus on node or graph classification tasks, graph auto-encoders (GAEs) used for anomaly detection present unique challenges. In this work, we aim to bridge this gap by formalizing the problem of explaining anomaly detection in GAEs and introducing methods that use reconstruction errors as explanations.

Some recent works [6, 27] have applied reconstruction errors as an explanation mechanism to guide experts in decision-making processes. However, these approaches lack a framework for comparison with other state-of-the-art explanation methods. To the best of our knowledge, no prior work has formally compared explainers derived directly from reconstruction errors to other explainability methods. Moreover, existing surveys on auto-encoder explainability [59, 14] primarily rely on model-agnostic explainers without

leveraging the reconstruction error as a source of explanation.

In this chapter, our key contributions are:

- We formalize the problem of explaining anomaly detection in GAEs, a previously unexplored area.
- We propose a novel method that leverages the reconstruction errors produced by GAEs to generate intuitive and interpretable explanations for anomalous nodes.
- We conduct a thorough comparison between our approach and state-of-the-art explainers, demonstrating the efficacy of reconstruction errors as a natural explainer for GAEs.

2.3 Definitions and Problem Formalization

As formalized in Section 2.1.3, we consider $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ an attributed network defined by the set of nodes $\mathcal{V} = \{v_1, \ldots, v_n\}$, the set of edges $\mathcal{E}$ represented by a symmetric adjacency matrix $\mathbf{A} = (a_{i,j}) \in \{0,1\}^{n \times n}$ where $a_{i,j} = 1$ if there is an edge between the nodes i and j and $a_{i,j} = 0$ otherwise, and the attribute matrix $\mathbf{X} = (x_{i,j}) \in \mathbb{R}^{n \times d}$ which ith row $\mathbf{x}_i$ represents the attribute vector of v_i. In the same way, $\mathbf{a}_i$ denotes the ith row of $\mathbf{A}$. In the following, bold upper case letters denote matrices, and bold lower case letters denote vectors.

Given a model f which produces a prediction $\hat{y} = f(v_i, G)$ for a node v_i, in the more general case, an explanation of this prediction is a vector $\mathrm{Imp}(f, v_i)$ which contains an importance weight for each node, edge, and attribute of the graph. Higher weights are associated with the most important elements for the prediction $f(v_i, G)$.

However, these importance weights for $f(v_i)$ are generally limited to elements (edges, nodes, and their attributes) located in the h-hop neighborhood

subgraph of the node v_i. This is the induced subgraph of G by all the nodes at a distance less or equal to h from v_i. It is denoted as:

$$\text{SUB}(v_i, h) = (\mathcal{V}_{\text{SUB}}(v_i, h), \mathcal{E}_{\text{SUB}}(v_i, h), X_{\text{SUB}}(v_i, h)). \tag{2.15}$$

Also, we split the importance weights into three vectors: $\text{Imp}_{\mathcal{V}}(f, v_i)$ is the importance vector with one weight for each node of $\text{SUB}(v_i, h)$, $\text{Imp}_{\mathcal{E}}(f, v_i)$ the weights for the edges of $\text{SUB}(v_i, h)$ and $\text{Imp}_{\mathbf{X}}(f, v_i)$ the weights for the attributes of node v_i. In the following, for simplicity, we will drop h and f from the notations.

We extend this problem formalization to a graph auto-encoder GAE (Figure 1.15) that takes as input the attribute and adjacency matrices $\mathbf{X}$ and $\mathbf{A}$ of an attributed graph G, compresses them into an embedding and then tries to reconstruct them, outputting the reconstructed matrices $\hat{\mathbf{X}}$ and $\hat{\mathbf{A}}$. The reconstruction error of each node is computed as follows:

$$error(v_i) = (1 - \alpha)\|\mathbf{a}_i - \hat{\mathbf{a}}_i\|_F^2 + \alpha\|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_F^2. \tag{2.16}$$

Nodes that obtain a high reconstruction error are considered anomalies.

The problem we seek to solve is to produce an explanation for each node identified as anomalous by a given graph auto-encoder.

Example: In a graph generated by ShapeGGen, nodes are assigned to one of two classes based on the number of specific subgraph structures, called motifs, within their 1-hop neighborhood. A node is labeled as normal (class 0) if it has at most one motif in its neighborhood and as anomalous (class 1) if it has more than one motif. Additionally, each node is given an attribute vector composed of several features. Among these, a few are considered informative, meaning they encode information specific to the node's class through distinct distributions. Both normal and anomalous nodes share the same informative features, but these features have different distributions depending

on the node's class, with a parameter that controls how much overlap there is between each distribution. The remaining features in the attribute vector are non-informative and do not contribute to the class distinction.

When the graph is analyzed, an effective explanation for why a node v_i is considered anomalous would be represented by two element masks, as designated by ShapeGGen. The first is an edge mask, which highlights all the edges within the motifs in v_i's 1-hop neighborhood, emphasizing the unusual presence of multiple motifs. The second is an attribute mask, which identifies the informative features whose distribution makes v_i stand out as an anomaly. These masks provide a clear and concise explanation by focusing on both the structural complexity and the distinct feature distributions that characterize the anomaly.

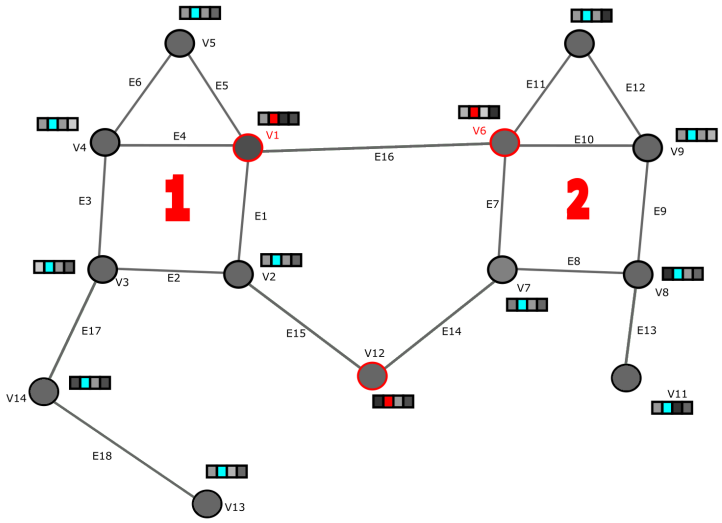


Figure 2.7: Example of a graph generated by ShapeGGen using houses as the motifs with three anomalies circled in red

In the graph presented in Figure 2.7, the edge explanation mask for v_1 would contain the edges: E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11,

E12, E16. The attribute explanation mask would be 0, 1, 0, 0, designating the informative feature.

2.4 Generating Explanation from the Reconstruction Errors of Graph Auto-encoders

2.4.1 Intuition

Our goal is to extract an explanation when an auto-encoder classifies a node as an anomaly. Most explanation mechanisms try to extract the weight of each feature from the model. In the case of auto-encoders, the detection mechanism is based on the reconstruction error, which is used to calculate the final score. The perfect explanation for such a model would be the explanation of why the elements are badly reconstructed. However, we hypothesize that taking the individual error of each component is already a good approximation of the contribution of those components to the classification, providing us with an already available explanation for the anomaly class.

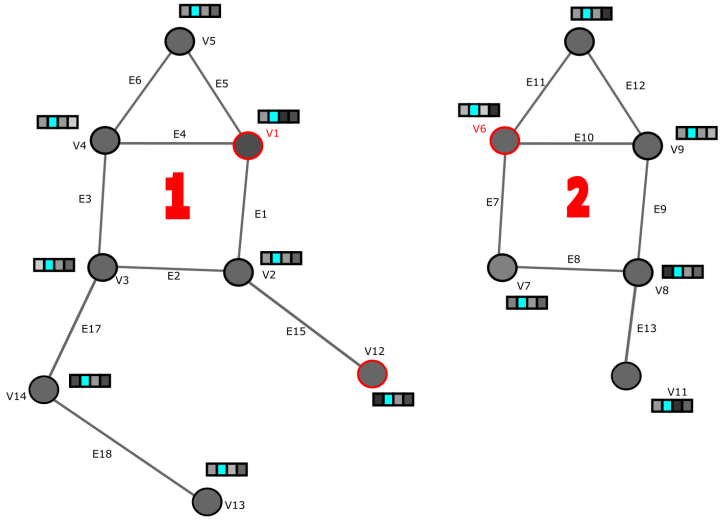


Figure 2.8: Example of a reconstruction of the graph from Figure 2.7 by a GAE

2.4.2 Importance Vectors

We propose a method to extract an explanation from the reconstruction error. We start by generating the error matrices $\mathbf{A}' = (a'_{i,j})$ and $\mathbf{X}' = (x'_{i,j})$:

$$a'_{i,j} = (a_{i,j} - \hat{a}_{i,j})^2, \; x'_{i,j} = (x_{i,j} - \hat{x}_{i,j})^2. \tag{2.17}$$

We can now use the reconstruction vectors $\mathbf{x}'_i$ and $\mathbf{a}'_i$ to create importance vectors, as they represent the contributions to the final classification. We can thus compute the feature importance vector $\text{Imp}_{\mathbf{X}}$ and the edge importance vector $\text{Imp}_{\mathcal{E}}$ as:

$$\text{Imp}_{\mathbf{X}}(v_i) = \mathbf{x}'_i, \; \text{Imp}_{\mathcal{E}}(v_i) = (a'_{j,k}, (v_j, v_k) \in \mathcal{E}_{\text{SUB}}(v_i)), \tag{2.18}$$

with $\text{Imp}_{\mathbf{X}}(v_i) \in \mathbb{R}^d$ and $\text{Imp}_{\mathcal{E}}(v_i) \in \mathbb{R}^{|\mathcal{E}_{\text{SUB}}(v_i)|}$.

Example: When an auto-encoder processes this graph, it tries to reconstruct each node's connections and attributes. Since the majority of nodes are normal (connected to at most one motif), the auto-encoder is good at reconstructing them. However, for the anomalous nodes with multiple motifs, the auto-encoder struggles, leading to higher reconstruction errors. These errors occur because the model defaults to reconstructing nodes as if they were normal, failing to capture the complexities of the anomalies.

If we consider a node v_i identified by our GAE. To explain why v_i was classified as anomalous, we examine the reconstruction error vectors:

- *The **edge importance vector** $Imp_{\mathcal{E}}(v_i)$ shows which edges in v_i's neighborhood were inaccurately reconstructed. In our example, the edges linking v_i to the multiple motifs should have high importance values as the model would try to reconstruct the node as connected to only one motif.*

- *The **attribute importance vector** $Imp_{\boldsymbol{X}}(v_i)$ would highlight the informative features of v_i, as they would be reconstructed as the majority class.*

For node v_1 in Figure 2.8, its edge importance vector would highlight the missing edge E16 the most, as the auto-encoder failed to reconstruct it. Its 1-hop edge importance vector could look like:

$$\text{Imp}_{\mathcal{E}}(v_1) = \{E1 : 0.15, E4 : 0.12, E5 : 0.11, E16 : 0.96\}$$

Additionally, the attribute importance vector would look like this:

$$\text{Imp}_{\mathbf{X}}(v_1) = \{x_1 : 0.12, x_2 : 1.25, x_3 : 0.34, x_4 : 0.23\}$$

With the highest value on the second attribute, which was incorrectly reconstructed for the normal distribution, the anomaly arose due to the difference in its true class distribution.

2.4.3 From Importance Vectors to Explanations

Most evaluation methods will require the transformation of importance vectors into binary explanation vectors such that $\text{Exp}_{\mathcal{V}}(v_i) \in \{0,1\}^{|\mathcal{V}_{\text{SUB}}(v_i)|}$, $\text{Exp}_{\mathcal{E}}(v_i) \in \{0,1\}^{|\mathcal{E}_{\text{SUB}}(v_i)|}$ and $\text{Exp}_{\mathbf{X}}(v_i) \in \{0,1\}^{d}$. In the protocol of [1], a fixed threshold of $0.8 \times max(\text{Imp}(v_i))$ is used, where $\text{Imp}(v_i)$ is an importance vector generated by the explainer (it can be either $\text{Imp}_{\mathcal{V}}(v_i)$, $\text{Imp}_{\mathcal{E}}(v_i)$ or $\text{Imp}_{\mathbf{X}}(v_i)$), $\max(\text{Imp}(v_i))$ corresponds to its maximum component and

$$\text{Exp}(v_i)[j] = \begin{cases} 0, \text{if } \text{Imp}(v_i)[j] < 0.8 \times \max(\text{Imp}(v_i)) \\ 1, \text{if } \text{Imp}(v_i)[j] \geq 0.8 \times \max(\text{Imp}(v_i)) \end{cases} . \tag{2.19}$$

Example: Consider node v_1 from the previous example, where the edge importance vector was:

$\text{Imp}_{\mathcal{E}}(v_1) = \{E1 : 0.15, E4 : 0.12, E5 : 0.11, E16 : 0.96\}$

Here, the maximum value in the vector is 0.96 *for edge* $E16$*. Using the threshold of* $0.8 \times 0.96 = 0.768$*, we transform the importance vector into a binary explanation vector:*

$\text{Exp}_{\mathcal{E}}(v_1) = \{E1 : 0, E4 : 0, E5 : 0, E16 : 1\}$

This binary explanation vector highlights that only edge $E16$ *has an importance value above the threshold and is therefore considered part of the explanation.*

Similarly, for the attribute importance vector:

$\text{Imp}_{\mathbf{X}}(v_1) = \{x_1 : 0.12, x_2 : 1.25, x_3 : 0.34, x_4 : 0.23\}$

The maximum value is 1.25 *for attribute* x_2*. Using a threshold of* $0.8 \times 1.25 = 1.0$*, the binary attribute explanation vector becomes:*

$\text{Exp}_{\mathbf{X}}(v_1) = \{x_1 : 0, x_2 : 1, x_3 : 0, x_4 : 0\}$

Thus, the explanation for v_1 shows that attribute x_2 and edge $E16$ are the most important components that contributed to the node being classified as anomalous.

However, this method is arbitrary, allowing each explainer to give explanations of varying sizes (the size of an explanation is the number of 1 in Exp) while effectively, we would want explainers to give the correct size. This leads to difficulty in evaluating those explainers.

Indeed, some metrics, such as fid+ and fid-, are sensitive to the size of an explanation because an explanation of maximal size will always yield the best performances possible. Since these metrics evaluate the importance of elements based on their presence or absence, for fid+, the metric measures the change in the model's prediction when the important elements are removed. Removing all elements will result in a significant change in the model prediction, thus maximizing the fid+ score. Similarly, for fid-, the metric assesses the change in the model's prediction when only the important elements are retained and the rest are removed. If the explanation includes all elements, retaining these elements will preserve the model performance, thus maximizing the fid- score. As the explanation size grows, it becomes more probable that essential elements are included.

For this reason, and following the approach of [5], we propose to use the top$k$ function that outputs the indices of the top k values in its input on the importance vector $\text{Imp}(v_i)$:

$$\text{Exp}(v_i)[j] = \begin{cases} 0, if j \notin \text{top}k(\text{Imp}(v_i)) \\ 1, if j \in \text{top}k(\text{Imp}(v_i)) \end{cases} \tag{2.20}$$

Moreover, in anomaly detection, we are primarily interested in explaining the abnormality, as normality is often only implicitly defined by the absence of abnormal elements. Instead, focusing on anomalies and identifying what deviates from expected behavior, is the most efficient way to capture what constitutes normality. For this reason, we will compute those metrics only for anomalies.

2.5 Experimental Protocol

In this section, we describe the experimental protocol that we followed to evaluate the interest of the approach described in the previous sections to explain GAE anomaly detection using reconstruction errors.

2.5.1 Metrics

GEA, precision, and recall can be used to compare the explanation mask and the ground truth mask when this last one is available, whereas necessity and sufficiency can be used otherwise. These metrics are defined below by considering that Exp denotes an explanation vector produced by an explainer, and Gt is the corresponding ground truth. They all take a value between 0 and 1, with 1 being the best possible outcome and 0 the worst.

The Graph Explanation Accuracy (GEA) is defined in [1] as the Jaccard index between the ground truth mask and the explanation mask.

$$GEA(\mathrm{Gt}, \mathrm{Exp}) = \frac{TP(\mathrm{Gt}, \mathrm{Exp})}{TP(\mathrm{Gt}, \mathrm{Exp}) + FP(\mathrm{Gt}, \mathrm{Exp}) + FN(\mathrm{Gt}, \mathrm{Exp})}, \quad (2.21)$$

where TP (True Positive) is the number of indices that contain a one in both Gt and Exp, FP (False Positive) is the number of indices that contain a one in Exp and a 0 in Gt, FN (False negative) the number of indices that contain a one in Gt and a 0 in Exp.

We can also measure performances in terms of precision and recall:

$$Precision(\text{Gt}, \text{Exp}) = \frac{TP(\text{Gt}, \text{Exp})}{TP(\text{Gt}, \text{Exp}) + FP(\text{Gt}, \text{Exp})}, \tag{2.22}$$

$$Recall(\text{Gt}, \text{Exp}) = \frac{TP(\text{Gt}, \text{Exp})}{TP(\text{Gt}, \text{Exp}) + FN(\text{Gt}, \text{Exp})}, \tag{2.23}$$

The previous metrics can only be used for the synthetic datasets where a ground truth explanation is generated.

For most of the datasets where a ground truth explanation is not available, we follow the protocol introduced in [5], which defines a good explanation as both necessary and sufficient by using, respectively, the fidelity+ ($fid+$) and fidelity- ($fid-$) metrics.

Given a model f, $\hat{y}_{v_i} = f(v_i, G)$ is the prediction of f for a node v_i *i.e.* the probability that v_i is an anomaly calculated from graph data $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$. Given an explanation mask $\text{Exp}(v_i)$, $G \setminus \text{Exp}(v_i)$ denotes the graph G where all the edges and attributes corresponding to a 1 in $\text{Exp}(v_i)$ are hidden (replaced by 0). We also denote $G \cap \text{Exp}(v_i)$, the graph where the only available information is the edges and attributes corresponding to a 1 in $\text{Exp}(v_i)$ (all other edges and attributes are replaced by 0).

Then the necessity ($fid+$) measures the change in the prediction when the only information available to f is $G \setminus \text{Exp}(v_i)$ and the sufficiency ($fid-$) the change when the only information available to f is $G \cap \text{Exp}(v_i)$. To adapt these metrics to the output of our models we extend both Equations 2.12 and 2.13 as:

$$fid+ = (\hat{y}_{v_i} - \hat{y}_{v_i}^{\setminus \text{Exp}}) \text{ with } \hat{y}_{v_i}^{\setminus \text{Exp}} = f(v_i, G \setminus \text{Exp}(v_i)), \tag{2.24}$$

$$fid- = 1 - (\hat{y}_{v_i} - \hat{y}_{v_i}^{\text{Exp}}) \text{ with } \hat{y}_{v_i}^{\text{Exp}} = f(v_i, G \cap \text{Exp}(v_i)). \tag{2.25}$$

Although these two metrics seem relatively straightforward, they require some adaptation to GAE. The output of a GAE is not a probability or a class but an anomaly score. While [37] proposes to normalize the score to

this aim, we claim that what is really important is the variation of the *rank* of each score and not the variation in the score itself. We use *rank*, so that the performance metric fits the metrics used for the evaluation of the model, that is, the AUC score, recall@k, precision@k, all based on rank and not on the score itself. Thus, we propose to use:

$$\hat{y}_{v_i} = \frac{Rank(error(v_i))}{|V|} \tag{2.26}$$

where Rank is a function that associates a node with its position in the ordered list depending on its anomaly score $error(v_i)$ such that the lowest score has rank 1 and the highest has rank $|V|$.

2.5.2 Datasets

The experiments are conducted on six real-world datasets and three generated datasets:

- Cora and Citeseer are popular public network datasets [65]. In these graphs, each node is a scientific publication, and an edge represents the citation of another publication. The attributes correspond to the content of the publication, represented as a bag-of-words binary vector.
- In Photo, each node is a product, and an edge exists between two products if they are often purchased together. The attributes are also bag-of-words vectors.
- Datasets generated using ShapeGGen the generator described in 2.1.8.

These datasets fall into two categories, and their characteristics are given in Table 2.5.

Real world datasets with injected anomalies (I): Cora, Citeseer, and Photo are modified to obtain contextual and structural anomalies following the methodology proposed in [45]. Specifically, contextual anomalies are

created by replacing the attribute vector of a node with the most different node attribute vector out of a range of randomly selected nodes. Structural anomalies are created by adding edges to the graph to create cliques of size c. We create two versions of each dataset, each with only one type of anomaly being introduced, leading respectively to Data-con, including contextual anomalies, and Data-str, including structural anomalies. We inject 100 anomalies in each version, as indicated in Table 2.5.

Synthetic Dataset (S): We also use ShapeGGen with the following parameters: 100 for the number of features, 10 for the number of informative features d_i, 0.0022 for the probability of connection, and 15 for the class separation, creating two distributions with almost no overlap.

We then consider the nodes connected to two or more motifs as anomalous. The low probability of connection makes the anomalous nodes rare, and the high-class separation makes the informative attributes very salient between the two classes.

We use three graph motifs from [1] to create the Shape-datasets: ShapeCircle, ShapeHouse, and ShapeFlag. These motifs are defined as follows:

- **Circle**: This motif consists of four nodes arranged in a cyclic structure, where each node is connected to two others, forming a closed loop.
- **House**: This motif consists of five nodes. Two nodes form the base, and three additional nodes connect to form a triangular roof and a supporting structure resembling a house shape.
- **Flag**: This motif consists of six nodes. Four nodes form a linear "pole" connected sequentially, and two additional nodes branch out horizontally from the third and fourth nodes in the sequence, resembling a flag structure.

These motifs form the basis of the corresponding Shape-datasets.

Dataset	Nodes	Edges	Attributes	Anomalies	Anomaly Rate
Cora-con/str(I)	2708	5278	1433	100	0.04
Citeseer-con/str(I)	3312	4732	500	100	0.03
Photo-con/str(I)	7487	119043	745	100	0.01
ShapeCircle(S)	~ 1000	~ 2500	10	~ 100	0.1
ShapeHouse(S)	~ 1000	~ 2500	10	~ 100	0.1
ShapeFlag(S)	~ 1000	~ 2500	10	~ 100	0.1

Table 2.5: Characteristics of the datasets.

2.5.3 Baselines

A second aim of the experiments that we have carried out is to compare the performances in terms of explainability and time of the approach based on error reconstruction that we propose to those of the state-of-the-art methods. To do this, we consider the following classical explainers, whose general characteristics are given in Table 2.1.

SubGraphX (SubGx): SubGraphX iteratively removes the nodes and edges from the larger graph and observes the impact on the prediction. The nodes and edges that cause significant changes in the prediction are considered crucial and are retained in the final explanation subgraph [85].

GNNExplainer (GNNEx): GNNExplainer trains a model to predict $fid+$ and $fid-$ when removing nodes and edges. Features that significantly affect the prediction when removed or modified are considered important and are included in the explanation [83].

Integrated Gradients (IGEx): Define a baseline graph that represents a "neutral" state with minimal information. It then measures how predictions change with feature changes and then generates importance scores by accumulating the gradient effect along a path from the baseline to the actual features [69].

Gradients (GradEx): Measures node importance as the weight of each node after computing the output gradient with respect to the node feature [70].

GuidedBP: Focuses solely on the features that have an important effect on the output. Positive gradients indicate how much increasing the value of the feature would contribute to the output, aligning with the desired prediction. By eliminating negative influences, GuidedBP aims to provide a clearer picture of the features that influence the decision-making of the model [7].

Random Explainer (RandEx): In addition, we consider a trivial baseline where importance vectors are randomly generated (each value in the vector is iid).

These methods form the baselines against which the performance of our proposed method using the reconstruction error and denoted **Reconstruction error (ours)** is compared.

We use the implementations of explainers from [1] and Dominant to detect the anomalies, as it is the simplest graph auto-encoder (with the implementation of [45]). These explainers are designed to take the output of the last layer from a classification model as input. However, for GAE, the output is the reconstruction, and thus, we have to adapt the explainers for GAE.

2.6 Experimental Results

As the results obtained for the different datasets are very similar, we report only those corresponding to one dataset for each category. The other results are provided in the appendix.

2.6.1 Precision, Recall, and GEA on Synthetic Datasets

Feature Explanations: The average results, with standard deviations, computed over five runs in terms of GEA@k, Precision@k, and recall@k for feature explanations obtained on ShapeHouse are displayed in Figure 2.9. These results show that the proposed method (Ours), despite its simplicity, significantly outperforms all the baselines for explaining the model's predictions according to the three evaluation metrics on the three datasets

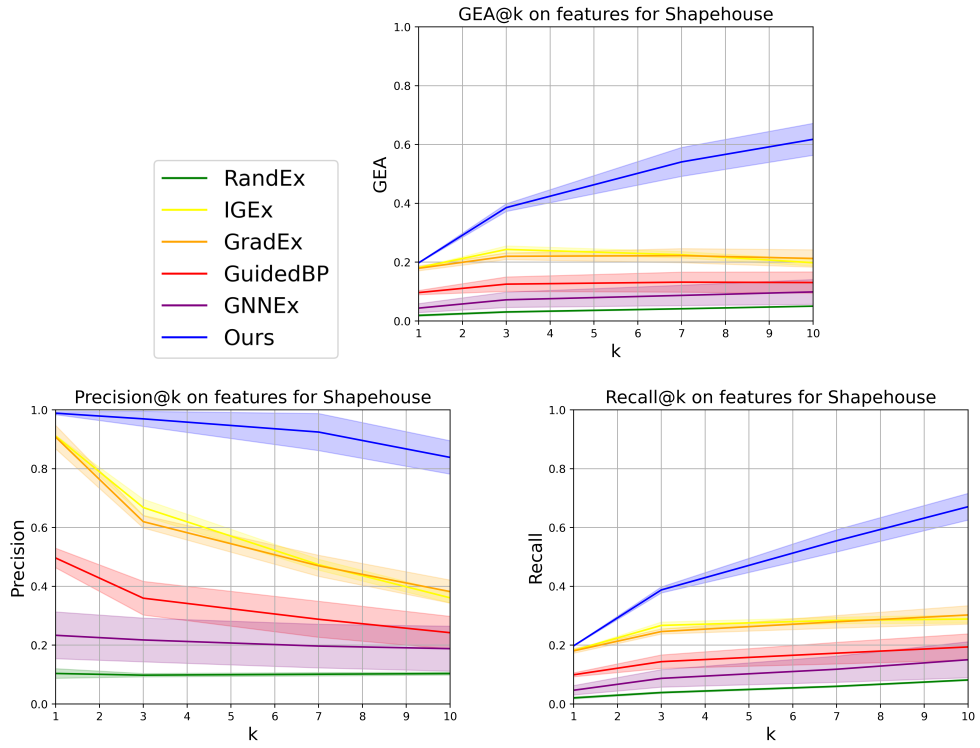


Figure 2.9: Average (+/- std) GEA@k (top), Precision@k (left), Recall@k (right) on features of ShapeHouse.

(cf. appendix for ShapeCircle and ShapeFlag). Integrated Gradients (IGEx) achieves the second-best score, performing similarly to our method for lower values of k but falling behind for higher values of k, providing only partial explanations. Grad (GradEx) shows performance equivalent to that of IGEx on ShapeFlag, but it had significantly lower performance on the two other datasets. Finally, the GNNExplainer is only slightly better than the random explainer (RandEx) for all metrics.

Edge Explanations: Figure 2.10 presents the average results, with standard deviations, computed over three runs according to GEA@k, Precision@k, and Recall@k for edge explanations on the ShapeHouse dataset.

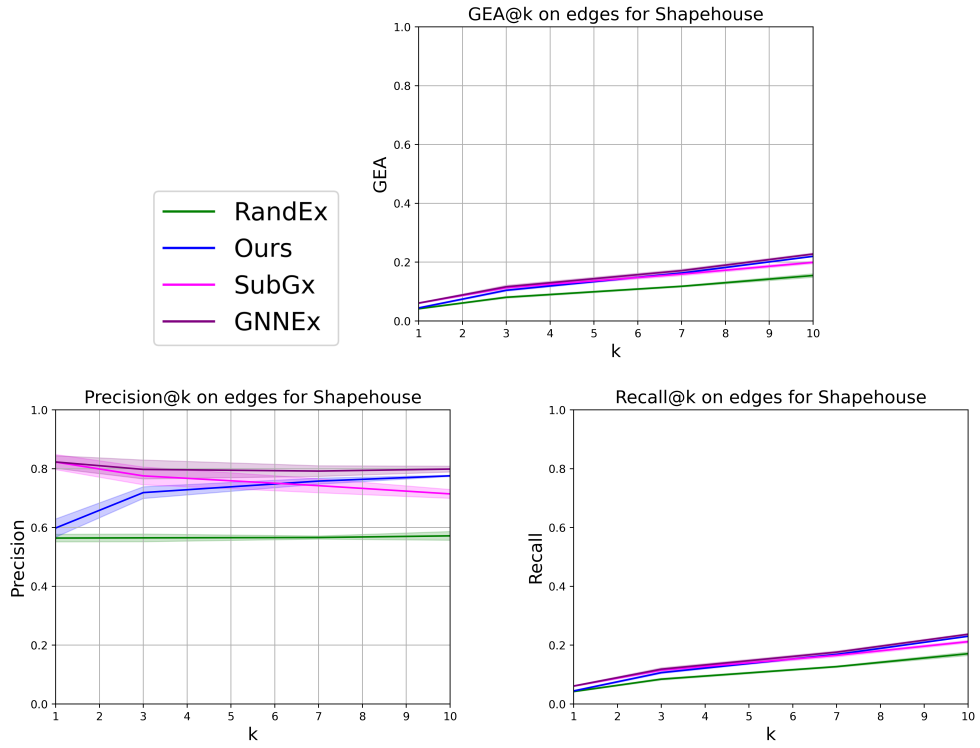


Figure 2.10: Average (+/- std) GEA@k (top), Precision@k (left) and Recall@k (right) on edges of ShapeHouse.

In this figure, we can observe that Ours, GNNEx, and SubGraphX achieve similar performances in terms of GEA@k and Recall@k and for $k > 2$ in terms of Precision@k.

2.6.2 Time Efficiency

The total time to compute all explanation masks for all anomalies on the three ShapeDatasets is presented in Figure 2.11 top (average and standard deviation over 10 runs). We can observe that SubGx takes considerably longer than all other explainers to generate those explanations. While every other method execution time is mostly influenced by the size of the sample

to explain, SubGx is also influenced by the number of elements in SUB(v_i, h) where v_i is the node to explain. Due to this, SubGx was excluded from the experiments on real-world datasets as they have a lot more edges, and its execution time increased to more than 10 minutes per node.

Similarly, Figure 2.11 (bottom) presents the total time for the real-world datasets. In this figure, we can observe that GradEx is the fastest method, with an average time of 12 seconds, followed by Ours with 18 seconds. However, this time difference can be attributed to the generation of edge explanations that GradEx does not produce. Afterward comes GuidedBp with around 25 seconds, followed by GNNex with an average of 32 seconds and a high standard deviation, and lastly IGEx with an average execution time of 57 seconds.

2.6.3 Average Necessity, and Sufficiency on all Datasets

Synthetic Datasets Figure 2.12 contains the average $fid+$ (left) and average $fid-$ (right) of the feature explanations computed over three runs on ShapeFlag. On this figure, we can see that using reconstruction error vectors as an explanation gives the best result on all three datasets and for both metrics (see appendix for ShapeHouse and ShapeCircle). Although only half of the performance of our proposed method was achieved, GradEx obtained the second-best results on all datasets. It is superior to all other explainers for both ShapeFlag and ShapeHouse and equivalent to GuidedBP for ShapeCircle (see appendix). GuidedBp and IGEx obtain relatively similar results. GNNExplainer shows the worst results in all tests while still performing better than the random explainer.

Figure 2.13 (top) contains the average $fid-$ and $fid+$ on edge explanation computed over three runs on ShapeCircle, and we report only one dataset, as the models behave exactly the same way across all datasets. We can observe that contrary to the previous figures, all explainers obtain the same results, an average $fid-$@0 of 1 and an average $fid+$@0 of 0. The $fid-$@0 of 1

means that the prediction without any edge is the same as with the edges. Thus, this shows that there are no necessary edges, meaning that the model does not use the structure at all, only the node attributes, for the predictions.

We have seen that IGex performed well in terms of GEA, Precision, and Recall. However, those results did not translate well in terms of necessity, and they fell behind those of GuidedBP, which previously outperformed IGex according to the previous metrics. This shows that the ground truth generated might be different from what the model actually detects. This observation is accentuated by the treatment of edges for those same datasets found in the top part of Figure 2.13, while most of the explainers showed great performance in terms of GEA, Precision, and Recall, $fid-$ and $fid+$ indicated that edges did not matter to the GEA. For edge explanation, the conclusion is the same and even worse as shown in Figure 2.13 (top): while the different models obtain good performances for edge explanation in terms of GEA, Precision, and Recall, it is not the case in terms of $fid-$ and $fid+$ which indicates that edges did not matter to the GAE. This divergence of the results according to the metrics demonstrates why it is important to capture different aspects of explainability: the reason why the node is an anomaly with GEA, Precision, and recall and why a model detects it as an anomaly with $fid-$ and $fid+$.

Real World Datasets with Injected Anomalies

Concerning the real-world datasets, we observed the same results for each type of anomaly, regardless of the original dataset. Thus, we will report the results over only one dataset for each type of anomaly. All the other results are available in the appendix.

Structural Anomalies. We report the average $fid-$@k and $fid+$@k obtained on Cora-str for the explanations of edges in Figure 2.12 (middle), and features in Figure 2.12 (middle). We observe a similar phenomenon as with

edge explanations for ShapeDatasets. Here, the features are likely not used by the models. Furthermore, while our method exceeds GNNEx in $fid-$ when explaining edges, all explanation methods achieve results inferior to RandEx. This signifies that currently, no method can successfully explain the decision of the model regarding this type of anomaly.

Contextual Anomalies. We report the average $fid-$@k and $fid+$@k on Citeseer-con for both edge explanations in Figure 2.13 (bottom) and feature explanations in Figure 2.12 (right). Since $fid-$@0 is not at 1 in both figures, we can conclude that for contextual anomalies, both edges and features hold value in explaining the model. For feature explanations, GradEx, GuidedBP, and Ours obtain equivalent results that are largely superior to other explainers with $fid-$ and $fid+$. GNNEx successfully obtains results that are better than those of RandEx, but it seems very unstable (high variance). IGex obtains poor performances that are equivalent to those of the random explainer. For edge explanation, all explainers obtain performance equivalent to the random explainer.

2.7 Conclusion

In this chapter, we introduced a comprehensive framework to evaluate explainability methods in the context of graph-based anomaly detection using GAEs (Graph Auto-Encoders). We adapted various state-of-the-art explainers from GNNs (Graph Neural Networks) to the GAE architecture and proposed a novel method to derive explanations directly from the reconstruction error vectors of GAEs. Extensive experiments on both synthetic and real-world datasets demonstrated that our proposed method of using reconstruction errors provides superior feature explanations, achieving the best or comparable performance across all six datasets according to the five metrics considered. For edge explanations, our method achieved results that

were equivalent to those of other explainers. Still, it highlighted a significant gap in current methodologies: the inability to provide complete and robust explanations for GAE-based anomaly detection consistently.

Our research indicates that using reconstruction errors as an explainer for GAEs is a promising approach but leaves significant room for improvement in the field of explainable AI for GAEs. Existing methods, including our own, performed poorly on real-world datasets when providing explanations for edge cases. The performance was similar to that of a random explainer. This emphasizes the need for future research to develop more effective explainers specifically designed to address the anomaly detection task in graph auto-encoder models.

In this chapter, we have focused on classical GAE models to explore the potential of reconstruction errors as explainers for anomaly detection, highlighting both their promise and limitations. In the next chapter, we shift our focus to *Suspicious*, our detection model introduced in Chapter 1, and apply it to a real-world application case. Additionally, we will demonstrate how our explainability approach can be effectively applied to this model, addressing the challenges of explaining its classification.

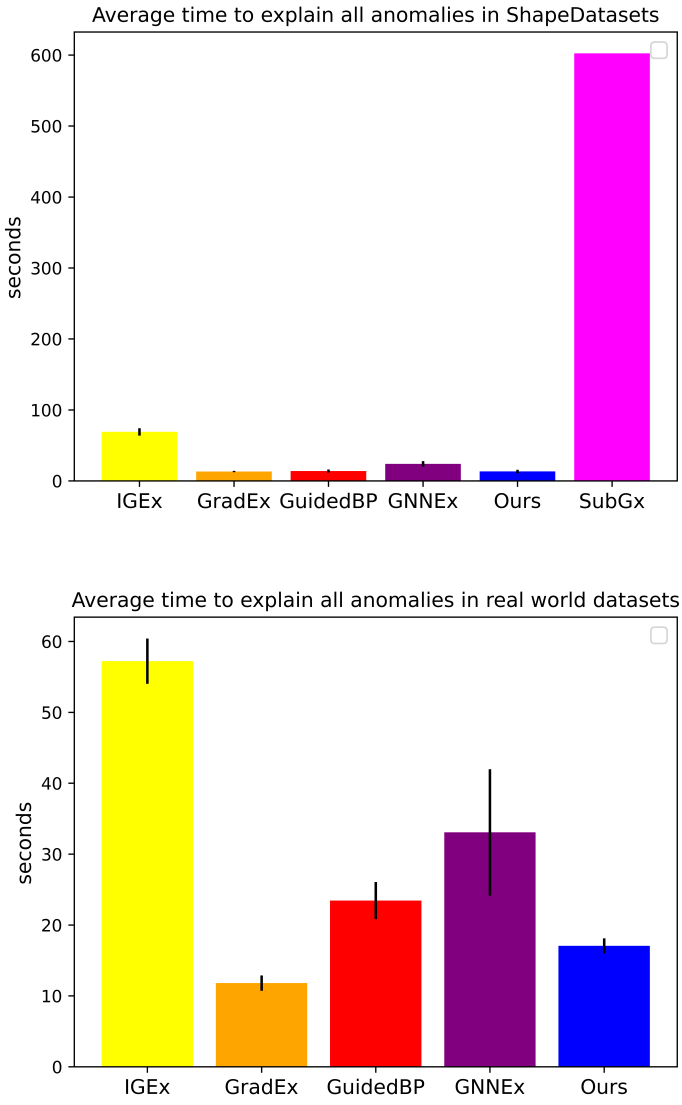


Figure 2.11: Average (+/- std) time to generate the explanation masks for ShapeDatasets (top) and real-world datasets (bottom).

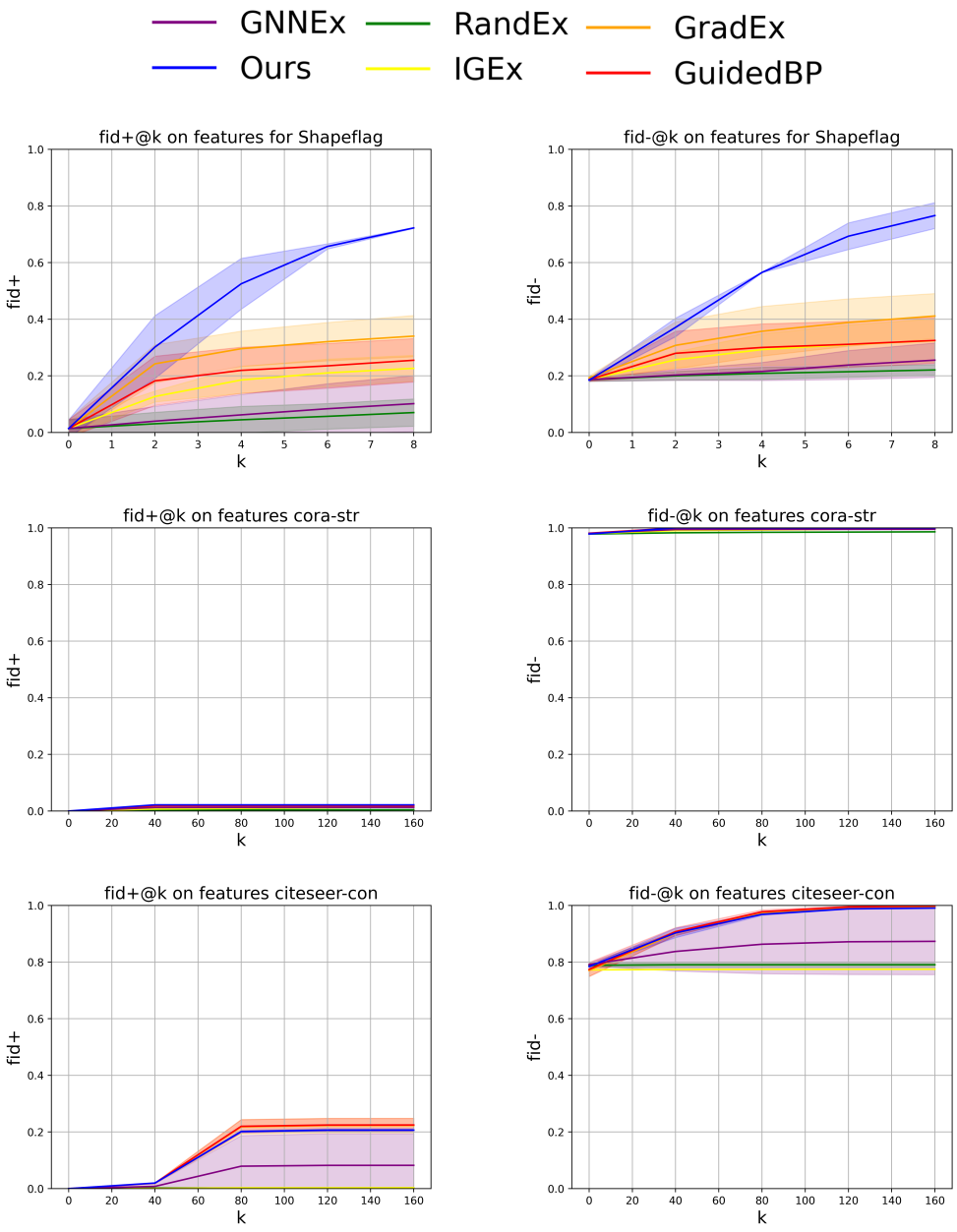


Figure 2.12: Average (+/- std) $fid+$@k (left), and $fid-$@k (right) on features of ShapeFlag (top), Cora-str (middle), Citeseer-con (bottom).

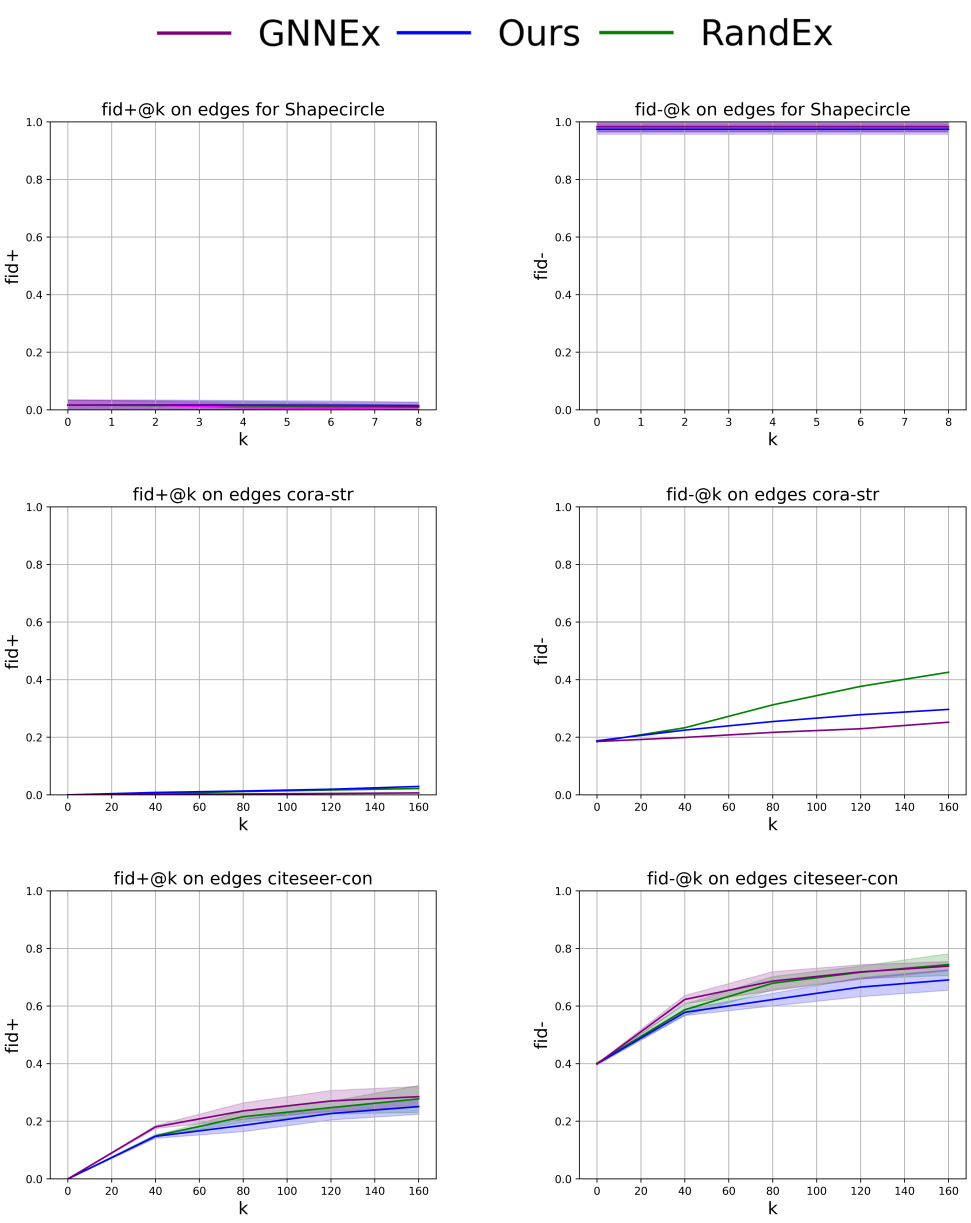


Figure 2.13: Average (+/- std) $fid+$@k (left), and $fid-$@k (right) on edges of ShapeCircle (top), Cora-str (middle), and Citeseer-con (bottom).

Chapter 3

Application Case

3.1 Application Case Presentation

This chapter delves into the specific application case of French health insurance reimbursement demands for glasses and lenses. This study focuses on detecting new and emerging patterns of insurance fraud within this dataset, a critical issue in the French healthcare system. Each reimbursement demand involves four key actors: the prescriber, the care provider, the beneficiary, and the insurer, as shown in Figure 3.1.

This study only concentrates on the optical specialty, i.e., glasses and lenses. The pipeline is as follows: when an insurance beneficiary needs glasses or lenses, he goes to his ophthalmologist (Prescriber). The ophthalmologist details the correction needed for the equipment in a prescription that the beneficiary then brings to his Optician (Provider). The optician follows the prescription and proposes various options, such as frames, to the beneficiary. This creates an insurance claim that is sent to the insurer for review. Beys, the company financing this thesis, receives this claim detailing the care selected and must validate it if it is legitimate or block it if it is detected as fraudulent. Our work represents the interaction with the insurer.

As detailed in the introduction, the company currently uses an expert

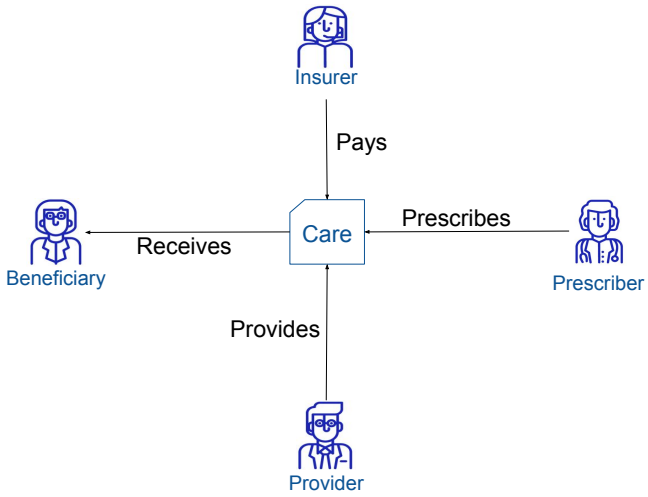


Figure 3.1: Interactions graph for a care.

system to detect fraud patterns identified by human experts. However, human experts cannot study the whole flux due to its size. By analyzing the interactions and data associated with these actors, we aim to identify known fraudulent activities and uncover new forms of fraud that may not have been previously detected. Here, new fraud refers to undiscovered fraudulent activities not yet identified by the current expert system, which models fraud patterns based on human expert insights.

We create detailed graphs from the dataset to encapsulate the relational information essential to this task. These graphs model the connections and interactions between the prescriber, care provider, and beneficiary, highlighting patterns that may indicate both known and novel fraudulent behavior. By leveraging graph mining in conjunction with real-world data, we aim to uncover novel fraud patterns, thereby enhancing the overall integrity of the healthcare reimbursement system.

3.1.1 Dataset Creation

Data Selection

Our only data source is the insurance claims filed by the provider, which contain identifiers for its actors and details on the care provided, such as detailed price distributions and specifications on what the care entails. In this work, we limit ourselves to optical claims. Even then, the company deals with a substantial number of health insurance claims annually. We focus on claims originating from Paris during April 2024 to create a manageable dataset. This selection results in 125,378 claims, each described by 69 attributes, including quantitative and qualitative features. Key attributes include the price of the glasses or lenses, the specific corrections prescribed, the prescriber's identification, the provider's details, and the beneficiary's demographic information.

Data Cleaning

Unlike publicly available datasets previously used, our dataset contains numerous errors due to poorly filled claims by the performers. The most common issues are unfilled attributes and duplicate claims. Addressing these issues involves a thorough data cleaning process, combining automated scripts and manual reviews to identify and rectify missing information and eliminate duplicate entries. Tools such as Python's `pandas` library are used for data manipulation and cleaning, ensuring the dataset's accuracy and reliability.

Graph Dataset

We created two versions of the graph dataset: First, a homogeneous graph containing only the claims and second, a heterogeneous graph containing nodes that can also represent providers, prescribers, and performers. Our work models the actions and interactions of other actors from the perspective of the insurer. As such, the insurer does not appear as a node in the

graphs.

Homogeneous graph: The first version is a homogeneous graph where each node represents a health insurance claim for a given care. The edges are established between two claims if they share a common actor: beneficiary, performer, or prescriber. The attribute matrix of this graph simply contains all the information of each claim as illustrated in Figure 3.2. This graph consists of 128,375 nodes, over 1 million edges, and 69 attributes. However, the various methods applied to this graph performed poorly, as detailed in section 3.2.1, which led us to develop a heterogeneous version.

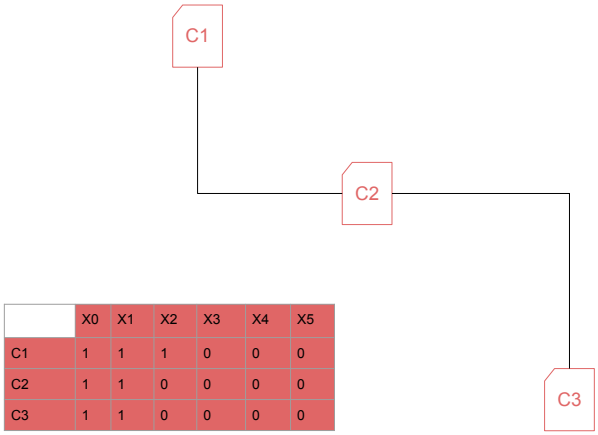
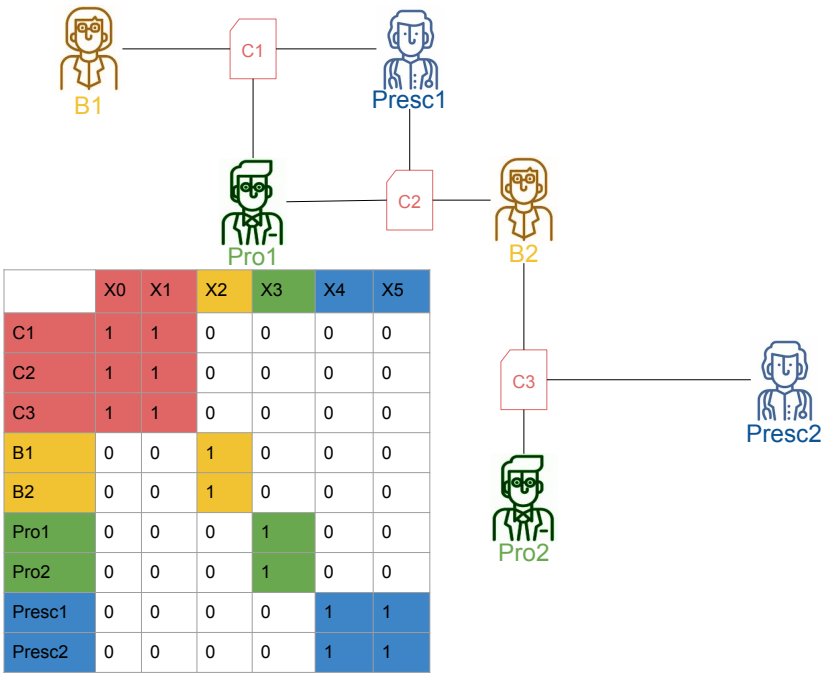


Figure 3.2: Homogeneous graph modelization with an example of an attribute matrix. Claims are linked to other claims when they share an actor.

Heterogeneous graph: In the second version, we created a heterogeneous graph by incorporating new types of nodes to represent prescribers and performers. The feature vectors for these new nodes are derived from their yearly statistical aggregation, as used by the expert system. This gives us a larger attribute matrix where each column describes features of only



	X0	X1	X2	X3	X4	X5
C1	1	1	0	0	0	0
C2	1	1	0	0	0	0
C3	1	1	0	0	0	0
B1	0	0	1	0	0	0
B2	0	0	1	0	0	0
Pro1	0	0	0	1	0	0
Pro2	0	0	0	1	0	0
Presc1	0	0	0	0	1	1
Presc2	0	0	0	0	1	1

Figure 3.3: Heterogeneous graph modelization with an example of an attribute matrix where the color designates the type of nodes and the column that contains their respective attributes. Red corresponds to claims, yellow to beneficiaries, green to providers, and blue to prescribers.

one node type. Each node contains its corresponding features, with zeros in all other features unrelated to its type. Each claim node is linked to its respective prescriber node, provider node, and beneficiary node, as shown in Figure 3.3. This produces a graph with 193,254 nodes, 382,125 edges, and 89 features.

3.2 Experiments

Our experiments are designed to evaluate the effectiveness of our models in detecting fraud through two primary approaches:

Methods	AUC±SD(%)
GAT classifiers(S)	**89.5** $\pm$ 3.2
GCN classifiers(S)	82.9 $\pm$ 3.5
Kumagai(S)	75.6 $\pm$ 2.4
Dominant(U)	49.7 $\pm$ 0.1
AnomalyDAE(U)	49.7 $\pm$ 0.1
Anomalous(U)	52.3 $\pm$ 0.0
Suspicious(S)	83.4 $\pm$ 4.2

Table 3.1: Average AUC±SD(%) over 10 runs obtained when we tried to use known frauds as the labels. The best score is 100, random is 50. Unsupervised method are denoted as U and supervised/semi-supervised methods as S

3.2.1 Detecting Known Frauds

Experimental Setup:

In the first approach, we use known fraud cases as labels for the supervised methods. We only use known frauds, i.e., detected by the expert system, as detection targets; for the supervised and semi-supervised methods, we provide 10% of the dataset as training, 10% as validation, and 80% as a test.

Experimental Results

We assessed how well our models could identify known fraud results for both types of graphs:

Homogeneous graph: In the case of the homogeneous graph, all methods, previously presented failed in this task, with all models stagnating around a 50% AUC score and no method reaching over 55%. The only types of fraud identified were those that could also be detected using only tabular data, such as:

> Claim with a very strong sight correction for a young child.

However, those fraud patterns are easily identifiable, which makes them rare. This suggests that this modelization does not allow the models to accurately capture the relationships in the graph, likely due to the extremely connected nature of the graph, where each node has an average of 14 neighbors, which is quite high compared to the graphs used in the experiments described in Chapter 1, where the average is around 3. Based on these results, we excluded this dataset from our future experiments.

Heterogeneous graph: The results for the heterogeneous graph are shown in Table 3.1. Most supervised methods such as GNN classifiers and *Suspicious* performed well, achieving an AUC score of around 80%. However, the elements detected by the unsupervised models performed nearly randomly, with a 50% AUC score.

While evaluating this approach is straightforward, it does not correspond to our task. We aim to uncover new and previously undiscovered frauds.

3.2.2 Detecting New Frauds

The second approach explores the model's ability to detect new and previously undiscovered fraud cases. Searching for undiscovered patterns means that no labels are available, and a fully human evaluation is required.

Experimental Setup:

In this second experiment, we do not have any labels as we aim to find undiscovered frauds.

To evaluate the results, we had to present the results of each model to our company's human experts. Due to the limited availability of experts, the process was split into two phases. In the first phase, we analyzed the model's output to present a small coherent ensemble of suspected frauds with the features that might have led to their categorization. In the second phase, the

experts investigated this ensemble to give the final verdict.

However, since this evaluation process is entirely human-led, it is prone to human error, making the evaluation long and challenging, greatly limiting the number of possible evaluations for each model.

Despite these challenges, we applied the best-performing methods used in previous sections to this new task and provided a detailed account of our findings in the following sections.

Unsupervised learning

The unsupervised method, Dominant [17], was the first contender we studied. As these methods do not use label information, we could use it directly on our datasets. Using this method on the heterogeneous graphs, we found already-known patterns corresponding to the notion of outliers. In the first version of these experiments, the top scores mostly corresponded to errors with abnormally filled fields such as the birth year, leading to extremely old beneficiaries (100 years old). This helped us better clean the dataset but did not provide insight into frauds. Once those instances were deleted and the model retrained on the new data, frauds such as overly priced glasses and late execution on prescriptions appeared as the top scores. These indeed corresponded to frauds, but the expert system had already identified these frauds. Deleting these instances and retraining the model did not lead to any new fraudulent or distinguishable pattern. This meant that while these models were still interesting, as they managed to identify known frauds without prior input, they failed to uncover any new fraudulent patterns. Other GAE methods, such as AnomalyDAE [19] or GUIDE [86], led to almost fully identical rankings and conclusions.

Supervised learning

We need labels to use semi-supervised learning methods. As shown in Section 3.2.1, providing already existing fraud leads the model to identify the same type of pattern, which does not allow us to identify new frauds. To provide the model with labels corresponding to new frauds, we used another learning model of the company that allows us to identify fraudsters, i.e., providers that commit fraud. Using this information, we used all of the claims from those suspects as the fraud learning sample for the model. The majority of the claims of these suspected providers are non-fraudulent since fraudsters do provide real cares. This method creates a very small sample that contains many non-fraudulent claims, with the new anomalies mixed into them. Using this, we hoped to find only the anomalies disproportionately present in the suspicious sample.

Through these methods, using supervised (e.g., GAT [74] and GCN [36] classifiers) and semi-supervised methods (e.g., Kumagai [38]), we did not find any evident patterns during step 1, even with the use of explanation methods. Experts could not distinguish any fraud in the proposed claims without this prior analysis.

Suspicious

Our method, *Suspicious*, identified three new fraud patterns. These patterns could be linked to already identified fraud patterns, but these specific instances were previously undetected by existing models. The first pattern was linked to familial fraud, i.e., false invoices for other family members of a recent client. Here, we observed a repeated fixed price that was not very high but always the same. Our explanation model based on reconstruction error accurately highlighted the specific price as the explanation for this fraud, showing an identifiable pattern previously unnoticed but not very widely used. The other two patterns identified by *Suspicious* were:

- *Instant distant collaboration*: This pattern involved distant collaboration where a provider sold glasses to several beneficiaries who had been prescribed glasses on the same day by a prescriber who worked 600 km away. Our model accurately highlighted both the distance and the date as the explanation for this pattern.

- *Price optimization*: A very frequent and already known pattern where the provider uses different techniques to try and find the maximal reimbursement sum to fix the price accordingly. In this instance, our model highlighted a yet undiscovered way those providers got this information that we will not disclose here, for confidentiality reasons.

The identification of these patterns by *Suspicious* demonstrates the model's ability to uncover new fraud schemes that were previously undetected. These findings have significant implications for improving fraud detection strategies and refining the expert system's models. However, the process is still challenging and requires extensive validation and continuous refinement to ensure the accuracy and reliability of the identified patterns. Additionally, GAE-based methods (Suspicious and Dominant) have shown the best results. However, the need for a dense adjacency matrix representation combined with a high number of claims requires a heavy amount of memory to function: around 24Go of RAM against the 4Go required by other methods. This limits the size of the dataset on which we can apply our method, forcing us to reduce it through restrictions in both the geographical sense (only Paris) and the duration (only a month). In comparison, the current expert system can cover the whole country with a 24-month history.

3.3 Conclusion

In this chapter, we presented a comprehensive analysis of French health insurance reimbursement demands for glasses and lenses, focusing on the detection

of both known and new fraud patterns. We detailed the dataset, including data selection, cleaning, and graph creation, and developed models to identify fraudulent activities. Our experiments employed both supervised and unsupervised learning methods to evaluate the effectiveness of these models.

Using existing fraud cases as labels, we found that semi-supervised and supervised learning techniques achieved high AUC scores, demonstrating strong performance in identifying known frauds. However, unsupervised methods failed to identify these existing frauds, underscoring the challenge of detecting new fraud patterns without labeled data.

However, to address the challenge of detecting new frauds, we employed a novel approach with our method, *Suspicious*, which successfully identified three new fraud patterns previously undetected by existing models. These patterns, including familial fraud, instant distant collaboration, and price optimization, highlight the potential of advanced graph-based models in uncovering sophisticated fraud schemes.

Despite the successes, the process of detecting new frauds remains complex and time-consuming due to the need for human evaluation. This human involvement is necessary because fraud cases often involve subtle patterns and evolving tactics that automated models may not fully capture. Our study's findings also emphasize the importance of continuous refinement and validation of fraud detection models to ensure their accuracy and reliability.

Overall, this study contributes to developing more robust and efficient fraud prevention strategies within the French healthcare reimbursement system, paving the way for future research and improvements in fraud detection methodologies.

Conclusion

This thesis explored the development and application of auto-encoder-based models on graph data to detect anomalies and fraudulent activities, particularly within the context of the French healthcare system's insurance reimbursement demands for glasses and lenses. The research aimed to address the challenges posed by the complexity and scale of real-world data, such as few labeled samples available and mislabeling errors, as well as the need for explainable and effective detection methods, it did so in three axis:

Suspicious

In the first part of this thesis, we introduced the graph anomaly detection task. We then proposed Suspicious, a comprehensive semi-supervised framework using GAEs. It was developed to detect anomalies in a more realistic setting, requiring very few examples and being resilient to mislabeling mistakes in that small sample. While Suspicious performed exceptionally well in these more challenging settings, it also showed strong results in classical cases, demonstrating its versatility. Experiments revealed that the concept of outliers used in unsupervised settings was too restrictive for real-world anomalies and that semi-supervised methods could not perform appropriately when we introduced mislabeling errors in the training sample. Suspicious, our proposed method overcame these restrictions.

Explaining Anomalies in Graphs

In the second chapter, we introduced an explanation method based on reconstruction errors, specifically designed to evaluate the explainability of GAEs in detecting anomalies. We compared this method with various other explainability techniques. The results showed that, while gradient-based methods are efficient, they often fail to provide complete explanations compared to advanced perturbation-based approaches. Our proposed method bridges this gap by offering explanations that are as high-quality as those provided by perturbation-based methods, while maintaining the time efficiency of gradient-based approaches. This research underscores that, although GAEs are powerful tools for anomaly detection, their explainability remains a significant challenge that must be addressed to enhance trust and transparency in AI systems.

Application to Real-World Fraud Detection

In the third chapter, we applied the *Suspicious* method and our reconstruction-error explainer to a real-world fraud detection case within the French healthcare system. We explored all previously presented fraud detection techniques by constructing detailed graph datasets that modeled the interactions between beneficiaries, prescribers, and providers. The results were varied: while supervised methods effectively identified known fraud patterns, unsupervised methods encountered difficulties. However, our *Suspicious* method successfully uncovered previously undetected fraud patterns, showcasing the potential of advanced graph-based models for real-world applications.

Contributions and Implications

This research makes several important contributions to the field of anomaly detection and fraud detection using graph-based models:

- **Suspicious:** A novel semi-supervised GAE-based framework for anomaly detection that outperforms other methods in both experimental and realistic settings.
- **An Explanation Method for GAEs:** An explainer that uses GAE's reconstruction errors to explain anomalousness.
- **Framework for Explainable GAEs:** The development of a comprehensive framework for evaluating the explainability of GAEs. Which provides a foundation for future research on making AI models more transparent and trustworthy.
- **Code Availability:** The full code for *Suspicious*, the explanation method, and the GAE framework is provided.
- **Identification of New Fraud Patterns:** The application of the *Suspicious* method to real-world data successfully uncovered new fraud patterns, highlighting the potential of graph-based models in operational settings. This achievement has practical implications for improving fraud detection strategies within the French healthcare system and other sectors.
- **Challenges and Limitations:** The study also highlighted several challenges, particularly the scalability of graph-based models and the need for more effective methods for detecting and explaining complex anomalies. These findings point to important directions for future research.

Future Works

- **Scalability and Efficiency:** Future work should focus on improving the scalability of graph-based models to handle larger datasets and longer time frames, enabling their application to broader and more complex real-world problems.
- **Enhanced Explainability:** There is a critical need to develop more sophisticated methods for explaining anomalies detected by GAEs, particularly in real-world datasets where the complexity of the data presents significant challenges.
- **Integration of Multi-Domain Data:** Expanding the application of graph-based models to incorporate data from multiple domains, such as other healthcare specialties, could provide more comprehensive insights and improve the effectiveness of fraud detection strategies.
- **Humans in Fraud Detection:** Reducing reliance on human evaluation by developing more automated tools could significantly enhance the efficiency and accuracy of fraud detection processes.

Acknowledgement

First and foremost, I would like to express my sincere thanks to Christine and Baptiste for their guidance and support throughout these past three years. Christine's broad vision and deep expertise in the field, along with her structured and rigorous approach, have been instrumental in shaping my research. Her expertise and clear direction were invaluable in helping me stay on track.

Baptiste provided crucial technical help, always willing to dive into the finer details and offer practical solutions. His assistance with the technical challenges we faced was instrumental in moving the work forward.

I am grateful to both Christine and Baptiste for their mentorship and collaboration during this journey.

I would also like to extend my gratitude to Damien, who provided constant guidance and support, particularly in navigating the application domain of my work. His insights were invaluable, and his encouragement throughout the process made a significant difference.

I am deeply grateful to Olivier for granting me the opportunity to embark on this thesis. His trust and support in the early stages were crucial in setting me on this path.

I would also like to extend my sincere thanks to the Hubert Curien Laboratory for providing an excellent research environment throughout my thesis. Their support and resources greatly contributed to the success of this work. I am equally grateful to Be-ys Research for financing this thesis and for their

trust in my work.

And finally, I would like to express my heartfelt gratitude to my friends and family for their unwavering emotional support and encouragement throughout this journey. Their presence and understanding were invaluable, providing me with the strength and balance needed to complete this work.

Bibliography

[1] Agarwal, C., Queen, O., Lakkaraju, H., Zitnik, M.: Evaluating explainability for graph neural networks. Scientific Data **10**(1), 144 (Mar 2023)

[2] Akoglu, L., McGlohon, M., Faloutsos, C.: Oddball: Spotting anomalies in weighted graphs. In: PAKDD. pp. 410–421 (07 2010)

[3] Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. Rev. Mod. Phys. **74**, 47–97 (Jan 2002). https://doi.org/10.1103/RevModPhys.74.47, `https://link.aps.org/doi/10.1103/RevModPhys.74.47`

[4] Allen, G.I., Gan, L., Zheng, L.: Interpretable machine learning for discovery: Statistical challenges and opportunities. Annual Review of Statistics and Its Application **11**(Volume 11, 2024), 97–121 (2024). https://doi.org/https://doi.org/10.1146/annurev-statistics-040120-030919

[5] Amara, K., Ying, Z., Zhang, Z., Han, Z., Zhao, Y., Shan, Y., Brandes, U., Schemm, S., Zhang, C.: Graphframex: Towards systematic evaluation of explainability methods for graph neural networks. In: Rieck, B., Pascanu, R. (eds.) Learning on Graphs Conference, LoG 2022. Proceedings of Machine Learning Research, vol. 198, p. 44. PMLR (2022)

[6] Assaf, R., Giurgiu, I., Pfefferle, J., Monney, S., Pozidis, H., Schumann, A.: An anomaly detection and explainability framework using convo-

lutional autoencoders for data storage systems. In: Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence. pp. 5228–5230 (2021)

[7] Baldassarre, F., Azizpour, H.: Explainability techniques for graph convolutional networks. CoRR **abs/1905.13686** (2019)

[8] Bandyopadhyay, S., N, L., Vivek, S.V., Murty, M.N.: Outlier resistant unsupervised deep architectures for attributed network embedding. In: Proceedings of the 13th International Conference on Web Search and Data Mining. p. 25–33. WSDM '20, New York, NY, USA (2020)

[9] Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., Herrera, F.: Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. Information Fusion (2020)

[10] Bo, D., Wang, X., Shi, C., Shen, H.: Beyond low-frequency information in graph convolutional networks. Proceedings of the AAAI Conference on Artificial Intelligence **35**(5), 3950–3957 (May 2021)

[11] Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: identifying density-based local outliers. In: ACM SIGMOD. pp. 93–104. ACM Press (2000)

[12] CAF: Bilan 2023 de la lutte contre les fraudes, `https://www.assurance-maladie.ameli.fr/presse/2024-03-28-dp-lcf`

[13] Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. **41** (07 2009)

[14] Charte, D., Charte, F., del Jesus, M.J., Herrera, F.: An analysis on the use of autoencoders for representation learning: Fundamentals, learning

task case studies, explainability and challenges. Neurocomputing **404**, 93–107 (2020)

[15] Chen, Z.X., Hohmann, L., Banjara, B., Zhao, Y., Diggs, K., Westrick, S.C.: Recommendations to protect patients and health care practices from Medicare and Medicaid fraud. J Am Pharm Assoc (2003) **60**(6), 60–65 (2020)

[16] Cybenko, G.V.: Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems **2**, 303–314 (1989)

[17] Ding, K., Li, J., Bhanushali, R., Liu, H.: Deep anomaly detection on attributed networks. In: SIAM ICDM. pp. 594–602 (2019)

[18] Dou, Y., Liu, Z., Sun, L., Deng, Y., Peng, H., Yu, P.S.: Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 315–324. ACM (2020)

[19] Fan, H., Zhang, F., Li, Z.: Anomalydae: Dual autoencoder for anomaly detection on attributed networks. In: ICASSP. pp. 5685–5689 (2020)

[20] FBI: `https://www.ccomptes.fr/fr/publications/la-lutte-contre-les-fraudes-aux-prestations-sociales`

[21] FBI: `https://www.fbi.gov/investigate/white-collar-crime/health-care-fraud`

[22] Fey, M., Lenssen, J.E.: Fast graph representation learning with pytorch geometric. ICLR (Workshop) (2019)

[23] Funke, T., Khosla, M., Anand, A.: Hard masking for explaining graph neural networks (2021), `https://openreview.net/forum?id=uDN8pRAdsoC`

[24] Gao, Y., Wang, X., He, X., Liu, Z., Feng, H., Zhang, Y.: Addressing heterophily in graph anomaly detection: A perspective of graph spectrum. In: Proceedings of the ACM Web Conference 2023 (2023)

[25] Giles, B., Jeudy, B., Largeron, C., Saboul, D.: Suspicious: a resilient semi-supervised framework for graph fraud detection. In: International Conference on Tools with Artificial Intelligence (ICTAI). pp. 212–220. Los Alamitos, CA, USA (nov 2023)

[26] Giles, B., Jeudy, B., Largeron, C., Saboul, D.: Un cadre semi-supervisé résilient pour la détection d'anomalie sur graphe attribué. In: Faron, C., Loudcher, S. (eds.) Extraction et Gestion des Connaissances, EGC 2023, Lyon, France, 16 - 20 janvier 2023. RNTI, vol. E-39, pp. 55–66. Editions RNTI (2023), http://editions-rnti.fr/?inprocid=1002810

[27] Gorman, M., Ding, X., Maguire, L., Coyle, D.: Anomaly detection in batch manufacturing processes using localized reconstruction errors from 1-d convolutional autoencoders. IEEE Transactions on Semiconductor Manufacturing **36**(1), 147–150 (2023)

[28] Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: ACM SIGKDD. p. 855–864. Association for Computing Machinery, New York, NY, USA (2016)

[29] Guyon, I., Gunn, S., Hur, A.B., Dror, G.: Design and analysis of the nips2003 challenge. In: Feature Extraction: Foundations and Applications, pp. 237–263. Springer, Berlin, Heidelberg (2006)

[30] Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NeurIPS. p. 1025–1035 (2017)

[31] Hawkins, D.M.: Identification of outliers. Monographs on applied probability and statistics, Chapman and Hall, London [u.a.] (1980)

[32] Hooi, B., Song, H.A., Beutel, A., Shah, N., Shin, K., Faloutsos, C.: Fraudar: Bounding graph fraud in the face of camouflage. In: SIGKDD. p. 895–904. KDD '16, Association for Computing Machinery, New York, NY, USA (2016)

[33] Huang, Q., Yamada, M., Tian, Y., Singh, D., Chang, Y.: Graphlime: Local interpretable model explanations for graph neural networks. IEEE Transactions on Knowledge and Data Engineering **35**(7), 6968–6972 (2023). https://doi.org/10.1109/TKDE.2022.3187455

[34] Huang, X., Yang, Y., Wang, Y., Wang, C., Zhang, Z., Xu, J., Chen, L., Vazirgiannis, M.: Dgraph: A large-scale financial dataset for graph anomaly detection. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems. vol. 35, pp. 22765–22777. Curran Associates, Inc. (2022)

[35] Interdonato, R., Atzmueller, M., Gaito, S., Kanawati, R., Largeron, C., Sala, A.: Feature-rich networks: going beyond complex network topologies. Appl. Netw. Sci. pp. 4:1–4:13 (2019)

[36] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)

[37] Kriegel, H.P., Kroger, P., Schubert, E., Zimek, A.: Interpreting and unifying outlier scores. In: SIAM International Conference on Data Mining. pp. 13–24 (Apr 2011)

[38] Kumagai, A., Iwata, T., Fujiwara, Y.: Semi-supervised Anomaly Detection on Attributed Graphs. In: IJCNN. pp. 1–8 (2021)

[39] Kumar, S., Zhang, X., Leskovec, J.: Predicting dynamic embedding trajectory in temporal interaction networks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery &

Data Mining. p. 1269–1278. KDD '19, Association for Computing Machinery, New York, NY, USA (2019)

[40] Li, A., Qin, Z., Liu, R., Yang, Y., Li, D.: Spam review detection with graph convolutional networks. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. pp. 2703–2711. ACM (2019)

[41] Li, J., Dani, H., Hu, X., Liu, H.: Radar: Residual analysis for anomaly detection in attributed networks. In: IJCAI. pp. 2152–2158 (2017)

[42] Lin, S.S., Jain, S., Wallace, B.C., Durrett, G.: Fidelity+ metric for evaluating the faithfulness of attention in natural language inference. In: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume. pp. 2555–2568 (2021)

[43] Liu, F.T., Ting, K., Zhou, Z.H.: Isolation-based anomaly detection. ACM Transactions on Knowledge Discovery From Data - TKDD **6**, 1–39 (03 2012)

[44] Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining. pp. 413–422 (2008)

[45] Liu, K., Dou, Y., Zhao, Y., Ding, X., Hu, X., Zhang, R., Ding, K., Chen, C., Peng, H., Shu, K., Sun, L., Li, J., Chen, G.H., Jia, Z., Yu, P.S.: Benchmarking node outlier detection on graphs. In: NeurIPS (2022)

[46] Liu, Y., Ao, X., Qin, Z., Chi, J., Feng, J., Yang, H., He, Q.: Pick and choose: A gnn-based imbalanced learning approach for fraud detection. In: Proceedings of the Web Conference 2021. pp. 316–327 (2021)

[47] Liu, Y., Li, Z., Pan, S., Gong, C., Zhou, C., Karypis, G.: Anomaly detection on attributed networks via contrastive self-supervised learning. IEEE TNNLS **33**(6), 2378–2392 (2022)

[48] Liu, Z., Cao, C., Sun, J.: Mul-gad: a semi-supervised graph anomaly detection framework via aggregating multi-view information. arXiv **2212.05478** (2022)

[49] Ma, X., Wu, J., Xue, S., Yang, J., Zhou, C., Sheng, Q.Z., Xiong, H., Akoglu, L.: A comprehensive survey on graph anomaly detection with deep learning. IEEE TKDE (2021)

[50] McAuley, J., Leskovec, J.: From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews. WWW 2013 - Proceedings of the 22nd International Conference on World Wide Web (03 2013)

[51] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. Proceedings of Workshop at ICLR **2013** (01 2013)

[52] Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. Artificial Intelligence **267**, 1–38 (2019)

[53] Molnar, C.: Interpretable Machine Learning. ., 2 edn. (2022)

[54] Muller, E., Sanchez, P., Mulle, Y., Bohm, K.: Ranking outlier nodes in subspaces of attributed graphs. In: ICDE. pp. 216–222 (04 2013)

[55] Peng, Z., Luo, M., Li, J., Liu, H., Zheng, Q.: Anomalous: A joint modeling approach for anomaly detection on attributed networks. In: IJCAI. pp. 3513–3519 (2018)

[56] Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: SIGKDD. p. 701–710 (2014)

[57] Pimentel, M.A., Clifton, D.A., Clifton, L., Tarassenko, L.: A review of novelty detection. Signal Processing **99**, 215–249 (2014).

https://doi.org/https://doi.org/10.1016/j.sigpro.2013.12.026, https://www.sciencedirect.com/science/article/pii/S016516841300515X

[58] Pourhabibi, T., Ong, K.L., Kam, B.H., Boo, Y.L.: Fraud detection: A systematic literature review of graph-based anomaly detection approaches. Decision Support Systems (2020)

[59] Ravi, A., Yu, X., Santelices, I., Karray, F., Fidan, B.: General frameworks for anomaly detection explainability: comparative study. In: 2021 IEEE International Conference on Autonomous Systems (ICAS). pp. 1–5. IEEE (2021)

[60] Rayana, S., Akoglu, L.: 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. KDD '15, Association for Computing Machinery, New York, NY, USA (2015)

[61] Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S.A., Binder, A., Müller, E., Kloft, M.: Deep one-class classification. In: ICML. vol. 80, pp. 4393–4402 (Jul 2018)

[62] Saeed, W., Omlin, C.: Explainable ai (xai): A systematic meta-survey of current challenges and future opportunities. Knowledge-Based Systems **263**, 110273 (01 2023). https://doi.org/10.1016/j.knosys.2023.110273

[63] Sanchez, P., Muller, E., Laforet, F., Keller, F., Bohm, K.: Statistical selection of congruent subspaces for mining attributed graphs. In: ICDM. pp. 647–656 (12 2013)

[64] Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Comput. **13**(7), 1443–1471 (2001)

[65] Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine **29**(3), 93 (2008)

[66] Shapley, L.S.: A value for n-person games. In: Kuhn, H.W., Tucker, A.W. (eds.) Contributions to the Theory of Games II, pp. 307–317. Princeton University Press, Princeton (1953)

[67] Shi, F., Cao, Y., Shang, Y., Zhou, Y., Wu, J., Zhou, C.: H2-fdetector: A gnn-based fraud detector with homophilic and heterophilic interactions. In: Proceedings of the ACM Web Conference 2022. pp. 1486–1494 (2022)

[68] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., Hassabis, D.: Mastering the game of go without human knowledge. Nature **550**(7676), 354—359 (October 2017). https://doi.org/10.1038/nature24270

[69] Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. In: ICLR (2014)

[70] Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. In: ICLR (2014)

[71] Tang, J., Hua, F., Gao, Z., Zhao, P., Li, J.: Gadbench: Revisiting and benchmarking supervised graph anomaly detection. In: Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S. (eds.) Advances in Neural Information Processing Systems. vol. 36, pp. 29628–29653. Curran Associates, Inc. (2023)

[72] Tang, J., Li, J., Gao, Z., Li, J.: Rethinking graph neural networks for anomaly detection. In: ICML. vol. 162, pp. 21076–21089 (2022)

[73] Tong, H., Lin, C.Y.: Non-negative residual matrix factorization with application to graph anomaly detection. In: SDM (2011)

[74] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. ICLR (Poster) (2018)

[75] Vu, M.N., Thai, M.T.: Pgm-explainer: probabilistic graphical model explanations for graph neural networks. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS '20, Curran Associates Inc., Red Hook, NY, USA (2020)

[76] Wang, X., Jin, B., Du, Y., Cui, P., Tan, Y., Yang, Y.: One-class graph neural networks for anomaly detection in attributed networks. Neural Computing and Applications **33**(18), 12073–12085 (2021)

[77] Wang, Y., Zhang, J., Guo, S., Yin, H., Li, C., Chen, H.: Decoupling representation learning and classification for gnn-based anomaly detection. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 1239–1248. ACM (2021)

[78] Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., Weinberger, K.: Simplifying graph convolutional networks. In: International Conference on Machine Learning. pp. 6861–6871. PMLR (2019)

[79] Wu, J., He, J., Liu, Y.: Imverde: Vertex-diminished random walk for learning imbalanced network representation. In: IEEE International Conference on Big Data (2018)

[80] Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: ICLR (2019)

[81] Xu, X., Yuruk, N., Feng, Z., Schweiger, T.A.J.: Scan: a structural clustering algorithm for networks. In: SIGKDD. pp. 824–833 (2007)

[82] Ying, R.Y., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: Gnnexplainer: Generating explanations for graph neural networks. In: Advances in neural information processing systems. vol. 32, pp. 9240–9251 (2019)

[83] Ying, Z., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: Gnnexplainer: Generating explanations for graph neural networks. In: Advances in Neural Information Processing Systems. vol. 32 (2019)

[84] Yuan, H., Yu, H., Gui, S., Ji, S.: Explainability in graph neural networks: A taxonomic survey. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(5), 5782–5799 (2023). https://doi.org/10.1109/TPAMI.2022.3204236

[85] Yuan, H., Yu, H., Wang, J., Li, K., Ji, S.: On explainability of graph neural networks via subgraph explorations. In: ICML (2021)

[86] Yuan, X., Zhou, N., Yu, S., Huang, H., Chen, Z., Xia, F.: Higher-order structure based anomaly detection on attributed networks. In: International Conference on Big Data (Big Data). pp. 2691–2700 (12 2021)

[87] Zeni, M., Zhang, W., Bignotti, E., Passerini, A., Giunchiglia, F.: Fixing mislabeling by human annotators leveraging conflict resolution and prior knowledge. Association for Computing Machinery **3**(1) (2019)

[88] Zhao, T., Deng, C., Yu, K., Jiang, T., Wang, D., Jiang, M.: Error-bounded graph anomaly loss for gnns. In: CIKM 2020. pp. 1873–1882 (10 2020)

[89] Zheng, Y., Jin, M., Liu, Y., Chi, L., Phan, K.T., Pan, S., Chen, Y.P.P.: From unsupervised to few-shot graph anomaly detection: A multi-scale contrastive learning approach. arXiv **2202.05525** (2022)