Catch Me If You Can: Graph Neural Networks to detect fraudulent nodes to counter money laundering

Master Project AI

Author: Maximiliane Ekert[2680863]
External Supervisor: Dr. Evert Haasdijk
Internal Supervisor: Dr. Michael Cochez
Second Supervisor: Dr. Xander Wilcke

Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands

Abstract. Money laundering poses a considerable threat to our society and thus countering money laundering has been included in the United Nation's Sustainable Development Report under the goal to maintain peace, justice, and strong institutions. The size of the problem calls for automated solutions and poses the challenge of identifying a relatively small amount of illicit accounts within a vast majority of licit ones. The goal of this research was two-fold: first, to investigate if graph-based methods are in general promising approaches in identifying illicit nodes within a transaction network and second, which of the implemented methods show the best results. As labels are scarce in practice and institutions try to avoid bias, we mainly focused on unsupervised methods. We compared stepwise approaches, i.e. first learning the node embeddings in an unsupervised manner using Deep Graph Infomax (DGI) with various encoders as well as node2vec which were then passed into a classifier with the goal to detect anomalies using an Isolation Forest (IF) or classify the nodes as illicit/licit with Logistic Regression (LR) and XGBoost (XGB). The stepwise approaches were contrasted to joined approaches combining these steps, i.e. DONE as anomaly detection algorithm and Graph Convolutional Networks (GCNs) for node classification. The results show that graph-based methods are in general suitable methods to identify illicit accounts above chance level with the best performance showing a stepwise approach of learning node embeddings with DGI using a transductive encoder and XGB as classifier. Furthermore, application of these methods in real-world settings poses additional challenges, specifically regarding the size of the data sets and privacy policiess.

Table of Contents

List of Figures

List of Tables

1 Introduction

The United Nations Office on Drugs and Crime (UNODC) estimates that every year between 2 - 5% of the global GDP is laundered, which roughly translates to $800 billion - $2 trillion annually [26]. In the Netherlands alone, €16 billion euros are laundered every year [89]. However, on average, only 1% of the illicit funds are detected [28]. Countering money laundering is of great societal interest and included in the 16th goal of the United Nation's Sustainable Development Report [85] advocating to maintain peace, justice, and strong institutions. The first global response to increasing incidents of money laundering was proposed in 1988 in the United Nation's Vienna Convention against narcotics and psychotropic substances (1988 Vienna Convention) [72] which defined money laundering as criminal act and nowadays forms the basis of the anti-money laundering (AML) laws and practices of most countries [25]. The subsequent year, the Financial Action Task Force (FATF) was established by the G7 countries which still sets global standards on AML policy today [29].

As the cases of AML are increasing, so have been the number of laws and regulations to which financial institutions have to comply to. If they fail to do so, high fines are issued that can in some cases even endanger profitability [100]. Financial institutions are thus inclined to prevent money laundering on their client accounts. Detecting the few cases of money laundering within a vast majority of licit accounts and transactions and considering the increasing amount of money laundering cases is no longer manageable with human labour alone, but calls for automated, smart solutions. Since criminals who launder money often work within a network, analysing the network of transactions and finding illicit behavior by taking the network structure into account seems to be a promising approach [100]. This research aims to investigate graph analytic techniques aimed at finding illicitly behaving account holders within the transaction network and thus seeks to contribute to the objective to countering money laundering.

1.1 Anti-money Laundering

The overall goal of money laundering is to gain capital by portraying money as legitimate assets that had been illegally obtained, i.e. whitewashing "dirty" money, and commonly follows three phases: placing the illicit funds (often cash) into the financial system (1. performance/placement), creating a complex trail of moving money around, e.g. through different countries (2. layering), and re-uniting the funds with the criminal, e.g. through purchases of luxury goods (3. integration) [107] (see also Figure 1). In an attempt to stay compliant with laws and regulations and to prevent criminally obtained money to enter the system, financial institutions have established a number of practices including "know your customer" (KYC; validating if the customer is who he/she claims to be), due diligence (continuous risk assessment of all customers of a financial institution, specifically for AML and counter-terrorist financing (CTF)) and transaction monitoring (checking behavior on an account to find any unusual or suspicious activity) [107].

Given the volume of customers, activities, and transactions, however, complying with the laws and regulations in place exceeds human capacities and calls for automated AML [107]. To prevent any false accusations of fraudulent behavior against clients which could jeopardizing the customer relationship or in the worst case lead to false convictions, human interference and checking of the outcomes of any applied methods is essential. Thus, AML is typically divided into three phases: initially transactions are analyzed electronically and alerts are generated, followed by a human expert assessing the validity of the alert(s). If one or more alerts cannot be disproven as false positives, the case must be reported to the local Financial Intelligence Unit (FIU) which links financial institutions to law enforcement [107]. One key aspect is the explainability of any raised alert(s) that can elucidate how risks have been assessed for the manual assessment, potential regulatory report thereof and for legal prosecution of involved parties to be successful.

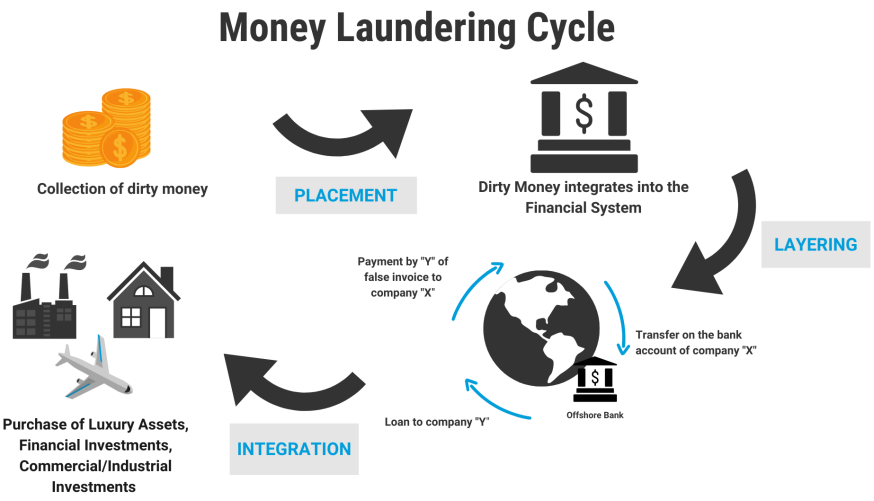


Fig. 1: Money Laundering Cycle showing the three phases of money laundering: 1. placement, 2. layering, 3. integration. Source: [26]

AML practices shifted from a rule-based approach, e.g. raising an alert if a transaction exceeds a fixed threshold of e.g. €10 000 to a risk-based one after too many false positive alarms had been raised. But as AML practices become more sophisticated, so do money laundering schemes: Criminals e.g. use "smurfing" to avoid any suspicion, i.e. splitting up money into smaller amounts which are subsequently distributed over multiple bank accounts or recruit "mules" that have no criminal history and whose accounts are used to channel funds [107]. Traditionally, financial institutions like banks counter money laundering individually. There are also attempts, however, to join forces and potentially raise the number of detected cases when banks share information amongst each other which oth-

erwise would only be available to each bank individually. The five Dutch banks ABN AMRO, ING, Rabobank, Triodos Bank and de Volksbank, for example, have established Transaction Monitoring Netherlands (TMNL) that combines the transaction data of companies holding an account at one or more of these partners [87].

1.2 Anomaly Detection in AML

Essentially, the goal of AML is to find the few illicit transactions/accounts within a vast majority of licit ones. This can be rephrased as finding the outliers where an outlier is "an observation point that is distant from other observations" [35]. Distance-, density-, classification- or cluster-based outlier detection techniques are usually applied to data represented in linear space [86], e.g. applying a anomaly detection algorithm while taking into account a number of a company's features. In real-world applications, however, objects might have meaningful relationships to one another which can provide valuable additional information [62]. Thus, not only account features, but also the graph structure that preserves information about which nodes are connected with one another can serve as important information and in some cases be the additional information needed to differentiate between fraudulent and normal users, e.g. in online social networks where fraudsters mimick attributes of benign users [5], [24], [110]. Preserving structural information of a network can lead to detecting fraudulent users or accounts which otherwise would have gone unnoticed [81].

Conventional anomaly detection methods make use of statistical models derived from domain expert knowledge, manual feature engineering or machine learning like matrix factorization or support vector machines (SVM) [4], [27], [54], [64]. These approaches, however, show shortcomings when it comes to scaling these models on real-world data set that have millions of nodes [75], [86]. Graph neural networks (GNNs) have recently gained popularity, amongst others, in AML [3], [18], [98], [110] and research targeting anomaly detection on graphs is broad [15], [19], [86]. A common approach is to use the data to first learn representations of the nodes that are depicted as vector features in the vector space, embeddings in the embedding space respectively, i.e. representation learning [12], [74], [75], [99]. Distributed embeddings are computationally efficient and widely applied in deep learning [41]. Anomaly detection algorithms are then applied to the vector space to detect outlying data points [75]. Isolation Forests (IFs), for example, are recently adopted anomaly detection algorithms that despite their simplicity show very promising results [6], [88]. However, not only unsupervised methods, but also supervised algorithms like Logistic Regression (LR) [13] can be applied as last step. This process can also be combined in joined graph anomaly detection algorithms with the goal to identify anomalous nodes, graphs or subgraphs within a graph or anomalous graphs amongst a set of graphs [10], [38], [100], [110]. Similarly graph classification algorithms for node classification (NC) serve as joined solutions. In either case, the nodes (V) of a (static, attributed) graph $G = \{V, E, X\}$ can intuitively reflect a bank account or a group of similar accounts with X representing attributes of the nodes, e.g. a

timestamp or an account ID, and the edges (E) the transactions between them [62], [100].

Although there is a growing research area investigating anomaly detection on financial data represented as graphs [98], most scientific studies leverage supervised or semi-supervised learning methods. Rao et al. [83], for example, apply a transaction prediction framework comprised of two parts: a detector based on a transformer architecture [91] and an explainer based on GNNExplainer [106] using heterogeneous graphs in a supervised learning setting for e-commerce. Van Belle et al. [10] and Weber et al. [100], [101] also leverage a supervised learning setup for credit-card fraud, fraudulent bitcoin transactions respectively comparing the performance of GraphSAGE [37] to Fast Inductive Graph Representation Learning [46] as well as contemporary methods to GCNs and combinations thereof taking space and time into account. Whereas Lv et al. [61] underline using auto-enconder-based GCNs to detect fraud, Liu et al. [59] and Liang et al. [55] propose stacking multiple adaptive path layers to select important neighbor nodes and aggregating the neighbors' features to detect insurance fraud. In order to bridge the gap between research and real-world applications, we selected a diverse set of algorithms which are applied to both, synthetic data, i.e. IBM AMLSim [100] as well as a data set consisting of real Bitcoin transactions [101].

1.3 Unsupervised representation learning

One of the central objectives of this research is to learn node embeddings without any prior knowledge, i.e. to learn representations of nodes in an unsupervised manner [41]. There are a number of models and frameworks that can be applied which can roughly be categorized in three different approaches [33], i.e. factorization based models like Laplacian Eigenmaps [9] which factorize the matrices representing the connections between nodes to obtain the representations (e.g. Radar [54]), random walk based techniques like DeepWalk [79] or node2Vec [34] that perform truncated random walks and maximize the probability of observing neighboring nodes on a random walk, and deep learning based approaches like structural deep network embedding (SDNE) [96] or variational graph auto-encoders (VGAE) [50] that apply methods used in deep neural network to graphs. In this research we will mainly focus on deep learning based approaches as well as a random walk based algorithm to compute node embeddings. In any case, the graph structure can be represented as an adjacency matrix $A \in R^{N*N}$ of N nodes with entries $A_{ij} = 1$ if a respective node pair is connected, and 0 otherwise [102]. Comparable to constructing node embeddings, creating suitable word embeddings have been motivated by the distributional hypothesis, which states that words that appear in a similar context have similar meaning [31]. On this basis, the Skipgram model, for example, represents words that have a similar context as vectors in the embedding space in close proximity to one another [67]. Similarly, the homophily hypothesis for graphs describes that connected nodes that are part of the same community should also be represented close to each other in the embedding space [32], [43]. The Skipgram model and homophily hypothesis

serve as basis and inspiration for more recent algorithms like the random-walk based techniques.

The different unsupervised representation learning algorithms can further be categorized in transductive versus inductive learning techniques which describe their capability to learn representations of unseen nodes [38]. Most models fall in the category of transductive learning techniques, i.e. the entire graph is needed during learning of the embeddings which causes challenges when encountering unseen nodes [37]. There are also a smaller number of inductive learning methods; Hamilton et al. [37], for example, proposed GraphSAGE, a framework and neural network layer that allows to induce the embeddings of unseen nodes based on their position in a graph (see also 2.2).

1.4 Challenges

There are a number of challenges when it comes to finding illicit nodes in graphs. First, graphs representing transactions over a longer period of time can include millions of nodes, as mentioned above, and thus call for scalable and efficient methods that can handle large graphs that continuously change [62], [83]. Furthermore, ground-truth labels are scarce in many real-world applications as well as labeled, publicly available data sets due to the privacy sensitive nature of fraud detection [10], [97]. Furthermore, the labels that are available in real-world data sets are usually defined by experts whose opinions might not always be in agreement with one another [62]. There is also a great imbalance between a high number of licit and a small percentage of illicit transactions which calls for suitable methods. Lastly, if a transaction or client is flagged as anomalous and an alert is created, human analyst further investigate the case and need an explanation which features led to this decision [62], [83], which is difficult to provide when using GNNs.

1.5 Objective

Countering money laundering and thus preventing whitewashing illicit funds is of great interest to keep society safe and to maintain trust in the financial system [85]. Financial institutions are being kept accountable for detecting money laundering on their clients' accounts, but large amounts of transactions and high false positive rates call for smarter, automated solutions. Transactions between different clients can be depicted as graphs and with the help of GNNs either nodes of the graph can be represented as vector features in the vector space, i.e. node embeddings in the embedding space [38] or joined solutions can be applied [62]. Anomaly detection algorithms like IF or supervised methods like Logistic Regression (LR) can further be applied on the embedding space to find illicit clients in the network, i.e. those who launder money. Alternatively, graph anomaly detection or NC algorithms can be applied to the graphs which combine these two steps.

The aim of the project at hand is twofold: first, we want to investigate whether graph-based algorithms are a suited method to differentiate between licit and

illicit nodes within a transaction network by applying a diverse collection of graph methods to simulated as well as real data. Specifically, we are interested if the implemented methods can generate useful node features or probabilities regarding the licitness of nodes. Secondly, we want to compare the performance of the different methods to one another, i.e. transductive versus inductive methods, supervised (node classification) versus unsupervised algorithms (anomaly detection) as well as joined approaches versus stepwise approaches (unsupervised representation learning + a non-graph-based machine learning algorithm). Given the scope of this project we did not take the dynamic nature of trasnaction graphs into account, e.g. [76], but only included static graphs. Neither did we specifically address explainabilty of the results and implemented methods, e.g. [106].

2 Methodology

In the following all applied models, encoders, and classifiers will be outlined, followed by a description of the experimental setup, the data sets used in this project, the node embeddings, and the applied evaluation procedures.

2.1 Models

Graph Convolutional Networks Convolutional neural networks (CNNs) have shown great success in many domains and have launched a new era of deep learning [52], [53], [105]. CNNs focus on input data in the Euclidean space, i.e. 2-dimensional, regular data like images, videos or text and are able to leverage local connections, shared weights, pooling, and apply multiple layers to learn local meaningful features.

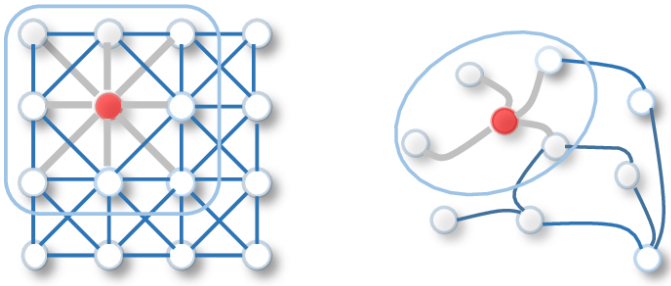


Fig. 2: 2D Convolution on the left vs. Graph Convolution on the right. Source: [105]

This type of data can also be viewed from a graph perspective: images, for example, can be represented as a regular grid in Euclidean space which abstractly can be understood as graph, but with a fixed structure. In this sense, pixels

represent nodes that are connected via edges to neighboring pixels (see also Figure 2).

As mentioned above, a growing body of research has been targeted towards non-Euclidean data. Deep learning paradigms, such as CNNs or autoencoders [94], for example, have been adapted to arbitrary graph structures resulting in Graph Convolutional Networks (GCNs), Variational Graph Autoencoders (VGAE) [50] respectively, which can produce powerful feature representations of graph nodes [23], [49], [62]. Comparably to 2D convolution in CNNs, graph convolution consists of summarizing the neighborhood information of a node by e.g. averaging (see also Figure 2). GCNs are originally designed for semi-supervised learning and need the entire graph Laplacian for training, i.e. are transductive [37], [49]. They are one of the most prominent deep graph learning techniques that serve as building basis for many other algorithms developed later in time.

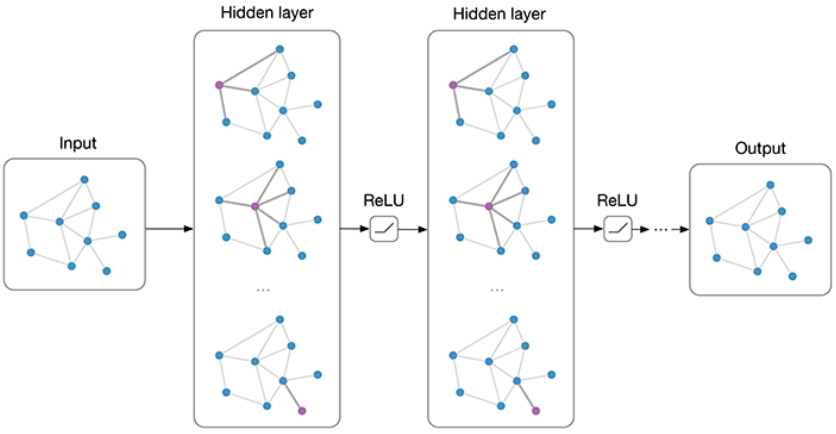


Fig. 3: Architecture of multiplayer Graph Convolutional Networks (GCNs) with input, hidden layers, each followed by a ReLU activation function and an output. Source: [49]

In particular, the feature representation of the next layer $H^{[l+1]}$ in a GCN is calculated by applying a non-linear activation function σ, e.g. the Rectified Linear Unit (ReLU) activation function [2], to the weights of the current layer $W^{[l]}$ times the feature representations of the current layer $H^{[l]}$ times the normalized adjacency matrix A^*: $H^{[l+1]} = \sigma(W^{[l]} * H^{[l]}) * A^*$ (see also Figure 3). A* implicating the normalized adjacency matrix using the diagonal node degree matrix (degree: amount of nodes a node i is connected to): $D^{-\frac{1}{2}} * AD^{-\frac{1}{2}}$ and the identity matrix: I to add self loops.

Each layer in a GCN convolutes the information of a node's neighbors that are one hop away. Thus, the number of layers indicates the number of hops the information can travel, e.g. with a three-layer GCN, each node's 'third-order' neighborhood is convoluted. When used for end-to-end classification as joined approach, GCNs' last layer is past through a softmax activation function:

$softmax(x_i) = \frac{1}{Z} exp(x_i)$ with $Z = \sum_i exp(x_i)$ applied row-wise. The error of the labeled data points are evaluated using the cross-entropy loss:

$$L = - \sum_{l \in Y_L} \sum_{f=1}^{F} Y_{lf} ln Z_{lf} \tag{1}$$

GCNs have continously been further developed. Wu et al. [104], for example, recently proposed the simplified GCN (SGCN) arguing that GCNs inherit redundant calculations and complexity that can be avoided. The authors removed all nonlinearities resulting in fewer parameters and significantly lower computation time while preserving aggregated information from the k-hop neighbors and scalability to large graphs and comparable accuracy scores.

Deep Graph Infomax Veličković et al. recently proposed their unsupervised representation learning framework Deep Graph Infomax (DGI) [93]. Due to its modular architecture and high performance compared to other popular unsupervised learning models we chose to use Deep Graph Infomax [93] as deep learning based unsupervised learning framework. DGI's modular architecture allows to easily exchange and test different implementations of its components, e.g. different encoders and thus e.g. transductive as well as inductive techniques can be used and compared to one another.

DGI is the adaption of Deep InfoMax (DIM) [42] to graphs, an unsupervised learning method of Euclidean data that trains an encoder model to maximize the mutual information between local areas of the input, e.g. patches of an image and a global representation of it. As such, DIM is strongly relatable to convolutional neural network structures. Similarly, DGI is based on maximizing mutual information between patch representations of the graph, $\overrightarrow{h_i}$ and high-level summaries of it [93] that are generated on the basis of established graph convolutional network architectures (see also Section 2.1).

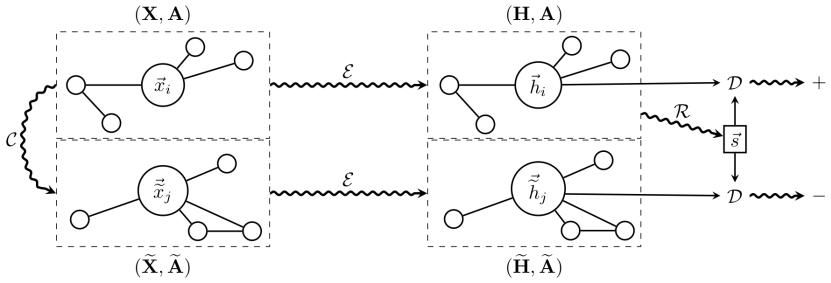


Fig. 4: Architecture of Deep Graph Infomax with the nodes of the real graph and its corrupted version (on the very left) being encoded (E) and scored against a summary vector of the real graph (s) using a discriminator (D) which is maximized for the real graph and minimized for its corrupted version. Source: [93]

Specifically, this framework's objective is to maximize local mutual information, i.e. to construct (local) node embeddings that represent (global) information of the entire graph which is captured in a summary vector $\overrightarrow{s}$. This summary vector $\overrightarrow{s}$ is created with the help of a read-out function that summarizes the patch representations into a graph-level representation: $\overrightarrow{s} = R(\varepsilon(X, A))$ with X: set of node features and A: the adjacency matrix. The read-out function can be as simple as a sum:

$$\overrightarrow{s} = \sigma\left(\frac{1}{N}\sum_{i=1}^{N}\overrightarrow{h_i}\right) \tag{2}$$

DGI is a constructive method, a technique that has also been successfully adapted to learning representations of words [21], [68], [70], such as, for example, the Skipgram model [67]. As such, DGI leverages a scoring function that contrasts "positive samples" or patches taken from the real graph against "negative samples" or patches $\overrightarrow{\tilde{h}_i}$ of a corrupted version of the graph $(\tilde{X}, \tilde{A})$. In order to maximize the local mutual information, a discriminator $D(\overrightarrow{h_i}, \overrightarrow{s})$ is used that assigns probability scores to each patch-summary pair. For patches that are part of the summary the probability scores should be higher and vice versa. The corrupted graph (discriminator: $D(\overrightarrow{\tilde{h}_i}, \overrightarrow{s})$) is obtained via a corruption function: $(\tilde{X}, \tilde{A})$ = C(X, A), e.g. shuffling of all features and randomly deleting edges between nodes. DGI then applies classification to score the representations [34], [38], [50], [79]. Finally, following the approach of DIM, DGI maximizes mutual information between $\overrightarrow{h_i}$ and $\overrightarrow{s}$ by applying a binary cross-entropy loss between the joint, i.e. the positive samples and the product of marginals, i.e. the negative samples:

$$L = -\log D(\overrightarrow{h_i}, \overrightarrow{s}) - \log(1 - D(\overrightarrow{\tilde{h}_i}, \overrightarrow{s})) \tag{3}$$

The schema of DGI is summarized in the following and depicted in Figure 4. For each node and its one-hop neighbors (i.e. patch of a graph rather than a node itself), denoted as graph G a corrupted version $H = C(G)$ thereof is created by e.g. shuffling the node attributes and keeping the edges stable. Both graphs are then encoded (different encoders can be applied, e.g. GCN 2.1) $E(G)$, $E(H)$ respectively and the true graph G is summarized in a summary vector $\overrightarrow{s} = R(E(G))$. The encoded embedding vectors of both graphs are compared to the summary vector using a discriminator $D(\overrightarrow{v}, \overrightarrow{s})$ for $\overrightarrow{v}$ in $E(G)$, $E(H)$ respectively and combined in a loss function. The loss function tries to maximize $D(\overrightarrow{v}, \overrightarrow{s})$ if $\overrightarrow{v}$ in $E(G)$ and minimize $D(\overrightarrow{v}, \overrightarrow{s})$ if $\overrightarrow{v}$ in $E(H)$ using gradient descent.

DONE There are a number of joined graph anomaly detection algorithms; DONE [8], for example, is a recently developed method that detects anomalies based on their global, structural, and community anomaly scores. Due to its relative high performance compared to other unsupervised approaches as well as

the fact that DONE includes both, the structure and the node attributes it was selected as joined graph anomaly detection in this research. For each node the likelihood of this node either sharing similar attributes with nodes from different communities (o_i^a), or being connecting to other communities (o_i^s), or being part of a community structurally, whilst being associated with a different one based on the attributes (o_i^{com}) is computed. The higher the likelihood in one or more of these three areas, the higher the score of this node and the higher the chance this node will be anomalous. DONE is based on two separate autoencoders, one for the structure and one for the attributes. During training the reconstruction errors of the autoencoders are minimized ensuring that the representation of connected nodes is similar in the embedding space (see also homophily hypothesis 1.3). DONE's loss function is anomaly-aware and consists of five terms: structure and attribute reconstruction errors: L_{str}^{Recs}, L_{attr}^{Recs}, preserving homophily:L_{str}^{Hom}, L_{attr}^{Hom}, and restrictions to ensure attributes and structure complement each other: L^{Com}. They are defined as:

$$L_{str}^{Recs} = \frac{1}{N}\sum_{i=1}^{N} log(\frac{1}{o_i^s})||t_i - \hat{t}_i||_2^2 \tag{4}$$

$$L_{attr}^{Recs} = \frac{1}{N}\sum_{i=1}^{N} log(\frac{1}{o_i^a})||x_i - \hat{x}_i||_2^2 \tag{5}$$

with structure and attributes of node i stored in t_i and x_i and reconstructed as $\hat{t}_i$ and $\hat{x}_i$.

$$L_{str}^{Hom} = \frac{1}{N}\sum_{i=1}^{N} log(\frac{1}{o_i^s})\frac{1}{|N(i)|}\sum_{j\in N(i)} ||h_i^s - h_j^s||_2^2 \tag{6}$$

$$L_{attr}^{Hom} = \frac{1}{N}\sum_{i=1}^{N} log(\frac{1}{o_i^a})\frac{1}{|N(i)|}\sum_{j\in N(i)} ||h_i^a - h_j^a||_2^2 \tag{7}$$

with the learned node representation from the structure (h_i^s) and attribute autoencoder (h_i^a).

$$L^{Com} = \frac{1}{N}\sum_{i=1}^{N} log(\frac{1}{o_i^{com}})||h_i^s - h_i^a||_2^2 \tag{8}$$

The sum of these loss functions are minimized and the k nodes with the highest scores are detected as anomalies [62].

2.2 Node encoders

Besides GCN and SGCN (see section 2.1, a number of GNNs, outlined below, have been selected that served as encoders for DGI.

Graph Attention Networks Contrary to GCNs, Graph Attention Networks (GATs) [92], another transductive method, do not share the same weights across all one-hop neighbors of a node i, but assign different levels of importance to each neighbor by leveraging masked self-attention and multi-head attention [91]. GATs are nowadays one of the most popular GNNs and considered state-of-the-art [17].

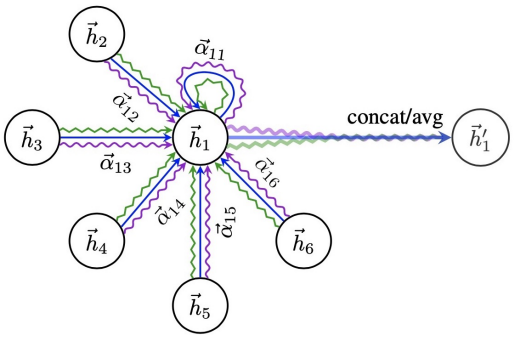


Fig. 5: Multi-head attention for Graph Attention Networks. [92]

First, a simple linear transformation is applied to every node h_i parametrized by their learnable weight matrix W of the current layer l to ensure that the input features can be transformed into higher level features: $z_i^{(l)} = W^{(l)} * h_i(l)$. Next, an attention coefficient is calculated for each pair of nodes i,j where a LeakyReLU activation function [63] is applied to the dot product between the concatenated z-embeddings of the two nodes $(z_i^{(l)} || z_j^{(l)})$ and a learnable weight vector $\overrightarrow{a}^{(l)^T}$:

$$e_i j^{(l)} = LeakyReLU(\overrightarrow{a}^{(l)\top} * (z_i^{(l)} || z_j^{(l)})) \tag{9}$$

In order to normalize all attention coefficiants of the one-hop neighbors of a node i, a softmax function is applied: $a_i j^{(l)} = \frac{exp(e_i j^{(l)})}{\sum_{k \in N(i)} exp(e_{ik}^{(l)})}$. Comparable to GCNs the feature representation of the next layer is calculated by aggregating the node embeddings of the neighbors $z_j^{(l)}$, however scaled by the attention scores $a_i j^{(l)}$: $h_i^{(l+1)} = \sigma(\sum_{j \in N(i)} a_i j^{(l)} * z_j^{(l)})$. In order to stabilize the learning and enrich the capacities of the model, GAT uses multi-head attention comparable to multiple channels in a CNN. A set number of independent attention mechanisms perform the last aggregation step and the resulting outputs can be either averaged or concatenated (see Figure 5).

Brody et al. [16] very recently have further improved GATs by expanding the attention mechanism from a *static* to a *dynamic* one resulting in improved performance and a more expressive method, i.e. GATv2. The authors propose to modify the order of GAT's operations and argue that in GAT's scoring function (see also Equation 9) the layers W and a can be collapsed into a single layer since they are consecutively applied. They proposed the following scoring function instead:

$$e_i j^{(l)} = \overrightarrow{a}^{(l)\top} LeakyReLU(W * [h_i^{(l)} || h_j^{(l)}]) \tag{10}$$

GraphSAGE As discussed above (see also Section 1.3), representation learning algorithms can be categorized in transductive and inductive methods. In order to deal with large and evolving graphs inductive methods are needed that do not need the full graph for training. Graph sample and aggregate (GraphSAGE) [37] leverages node attributes of neighboring nodes and shows a relatively high performance. It is based on the GCN architecture but instead of aggregating the information of all neighbors of a node, node embeddings are created by sampling and aggregating the features from a a fixed amount of a node's local neighbors (see also Figure 6. GraphSAGE ensures that connected nodes have similar node embeddings and non-connected ones distant ones (see also homophily hypothesis in Section 1.3) by applying the following loss function:

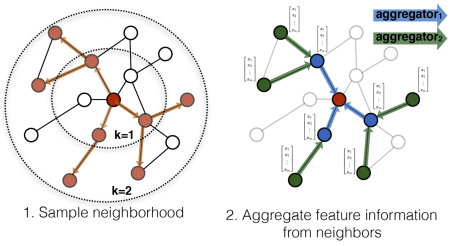


Fig. 6: GraphSAGE approach. [37]

$$J_G(z_u) = -\log(\sigma(z_u^\top z_v)) - Q * E_{v_n \sim P_n(v)} \log(\sigma(z_u^\top z_{v_n})) \tag{11}$$

with node v and u are near each other on a fixed-length random walk, the sigmoid function as σ, $P(n)$ representing a negative sampling distribution, and Q as the numbers of negative samples.

2.3 Classifiers applied on Node Embeddings

In case of the stepwise approach, classifiers targeting anomaly detection or classification are applied with the created node embeddings as input matrix. The selected algorithms, i.e. Isolation Forest, Logistic Regression, and XGBoost are outlined in the following.

Anomaly Detection Isolation Forests are relatively simple, but well performing anomaly detection algorithms. IFs are a distance-based tree ensemble method based on decision trees that isolate samples in the feature space by performing

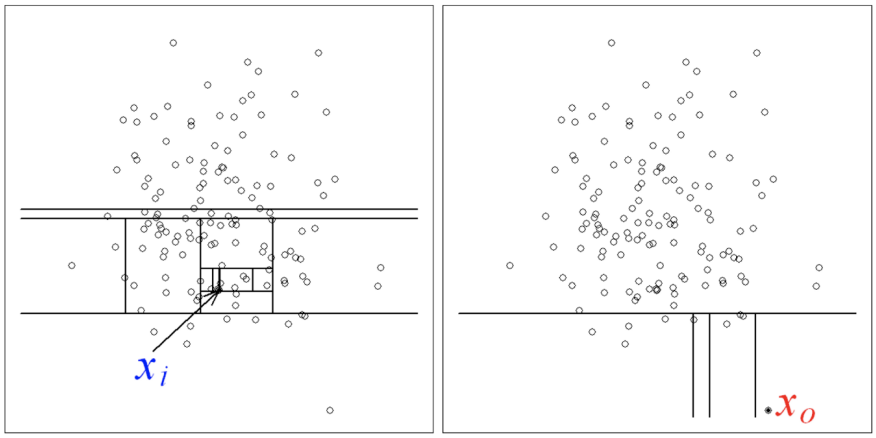


Fig. 7: Splitting for a normal versus anomalous data point. [56]

random partitions [56], [57]. A feature and a corresponding split value (between the minimum and maximum) are randomly selected to create partitions. A tree structure represents the recursive partitioning where the path length from the root to the termination node is equivalent to the total amount of splits needed to isolate a data point. Since anomalous data points lie further away from all other points in the feature space , they need fewer splits to be isolated and thus lie closer to the root of the tree, i.e. have shorter paths. An anomaly score is calculated for each sample x using its path length $h(x)$ averaged over a forest of n random trees $c(n)$:

$$s(x, n) = 2^{-\frac{E(h(x)}{c(n)}} \tag{12}$$

with scores close to 1 indicating anomalies and normal observations showing scores <0.5.

Classification In terms of supervised classification algorithms we implemented a simple Logistic Regression (LR) [13] algorithm as well as the more recently developed eXtreme Gradient Boosting (XGBoost) [20]. LR is one of the benchmark methods applied in anti-money laundering (AML) research [34], [69], [101], mainly due to its high explainability. XGBoost is a leading, state-of-the art machine machine learning algorithm and library that excels in regard to speed and performance. It is based on gradient-boosted decision trees and ensures parallel tree boosting. It uses a depth-first tree pruning approach and regularizations to avoid overfitting. XGBoost uses a logistic loss function:

$$L = \sum_i [y_i ln(1 + e^{-\hat{y}_i}) + (1 - Y_i) ln(1 + e^{\hat{y}_i})] \tag{13}$$

Van Belle et al. [10], [11] applied XGBoost for fraud detection in graph representation learning and found promising results for XGBoost applied to numeric vectors representing the graph structure.

2.4 Experimental Setup

This project was carried out in collaboration with Deloitte Financial Advisory B.V. and Transactie Monitoring Nederland (TMNL). It followed the cross-industry standard process for data mining (CRISP-DM) model [103]; a common and widely adopted approach to structure data science projects that entails six phases: business understanding, data understanding, data preparation, modelling, evaluation, and deployment [51], [103]. All steps were applied, but the last one as it exceeds the scope of this research project. The phases were in general followed chronologically; the different methods were first developed and evaluated on the synthetic data (see also 2.5) and only then applied to the real-world data set (see also 2.5).

Unsupervised: Anomaly Detection				**Supervised: Classification**			
Stepwise			**Joined**	**Stepwise**			**Joined**
Node Embedding		**Classifier**		**Node Embedding**		**Classifier**	
Transd.	**Inductive**			**Transd.**	**Inductive**		
DGI+	DGI+	IF		DGI+	DGI+	LR	
GCN	SAGE			GCN	SAGE	XGB	
SGCN			DONE	SGCN			GCN +
GAT	node2vec			GAT	node2vec		softmax
GATv2				GATv2			

Table 1: Overview over all implemented methods categorized as unsupervised vs supervised, stepwise vs. joined approach, for the stepwise approach the used node embeddings (transductive vs. inductive) and classifiers.

The code was developed in a Google Colaboratory [14] notebook and run on the corresponding servers. The GNNs were implemented using Pytorch Geometric [30], PyGOD respectively for DONE [58]. IF and LR were implemented using Scikit-learn [78] which was partially also applied during preprocessing and evaluation. The visualizations were done with the help of Matplotlib [45], for the graph NetowrkX [36] was used. For data handling Numpy [39], [73] and pandas [66] as well as tensorflow [1] and PyTorch [77] was applied.

The pipeline was implemented using the following hyperparameters: 50 epochs (convergence of loss curves), a learning rate of 0.01 (randomly chosen), 32 hidden channels (due to size of data sets rather small number chosen), and a 5-fold stratified cross validation (with shuffling) for LR (Scikit-learn) and XGBoost

(XGBoost library). The algorithms were trained with the Adam optimizer [48], but for node2vec SparseAdam [48] was applied as suggested by Pytorch Geometric (for both: torch.optim). The hyperparamters were not systematically tuned, as this would have exceeded the scope of this project. In order to ensure scalability the 'saga' solver was applied for LR with a 'balanced' class weight. For XGBoost the 'binary:logistic' objective was applied; for IF and DONE a contamination of 0.02 was used.

A diverse set of models was implemented (see also Table 1). All joined as well as stepwise approaches were implemented with 2 layers. As corruption procedure for DGI the node features were shuffled and edges were randomly removed; all encoders' hidden layers were passed through a PReLU activation function [40]. For the node2vec algorithm all default paramater settings were kept and a walk length of 20 with a context size of 10 and 2 walks per node was applied. For DONE the default parameter settings were used besides the globally set parameters (e.g. number of layers, learning rate, epochs, number hidden layers). For the joined GCN + softmax model a basic implementation with an output layer of size 2 and a ReLU [71] activation function inbetween layers was applied; the last layer was past through a sigmoid activation function.

2.5 Data sets

The above described techniques were applied to a simulated data set, i.e. AMLSim data set and a real-world data set, i.e. Elliptic data set. Ideally, we would have analyzed a real-world data set in a real-world setting, i.e. TMNL data set with the implemented methods, however due to a number of challenges outlined below that would have exceeded specifically the duration limitations of this project.

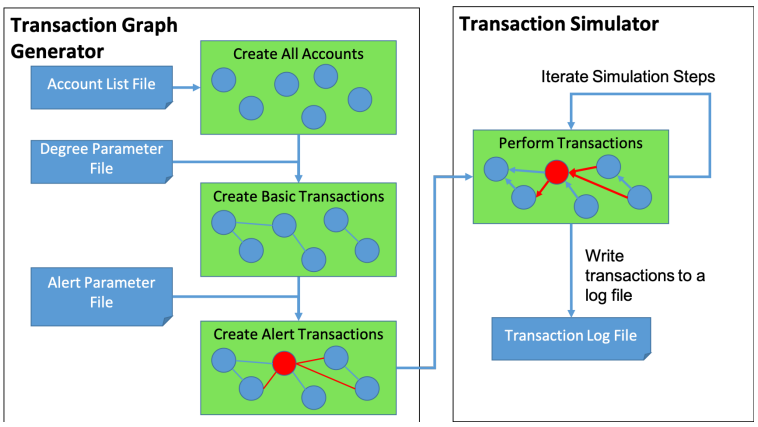


Fig. 8: Architecture of AMLSim simulation platform depicting its 2 components: the Transaction Graph Generator and the Transaction Simulator. Source: [100]

AMLSim data set AMLSim [100] is a multi-agent simulation platform trying to mimic real-world money laundering patterns where each agent represents a bank account that transfers money to other agents with a small percentage showing illicit behavior. The simulator has two components, a "Transaction Graph Generator" and a "Transaction Simulator" (see also Figure 8). The former first generates a graph based on a degree distribution using NetworkX and subsequently the "Transaction Simulator" generates transactions (time-series) using PaySim [60] which are based on transaction distributions and known money laundering patterns. For all nodes labels 'licit' or 'illicit' are added, 6.4% are labeled as 'illicit' accounts in this simulation. For this project, transaction data with ca. 10k nodes and 200k edges were simulated where each node represents an account and each edge a transaction between 2 accounts. In total there were 12,042 accounts and 199,374 transactions. Figure 9 depicts the transaction graph for 1 timepoint.

Preprocessing From the original node features, we only kept the "initial balance" of each account and deleted all other features (e.g. date of birth, address). Regarding the transactions all columns, but the ones describing sender and receiver account, transaction amount, timestamp, and whether a transactions is fraudulent or not were deleted. The timestamp was converted into a date format and encoded as labels (LabelEncoder from Scikit-learn). Data formats were adjusted for all remaining node and edge features and the columns were renamed to sensible descriptions. There was no missing entries found in the data set.

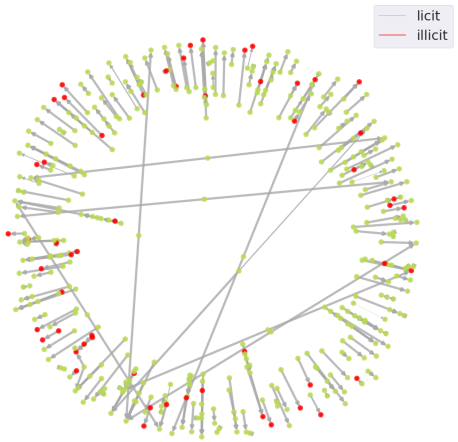
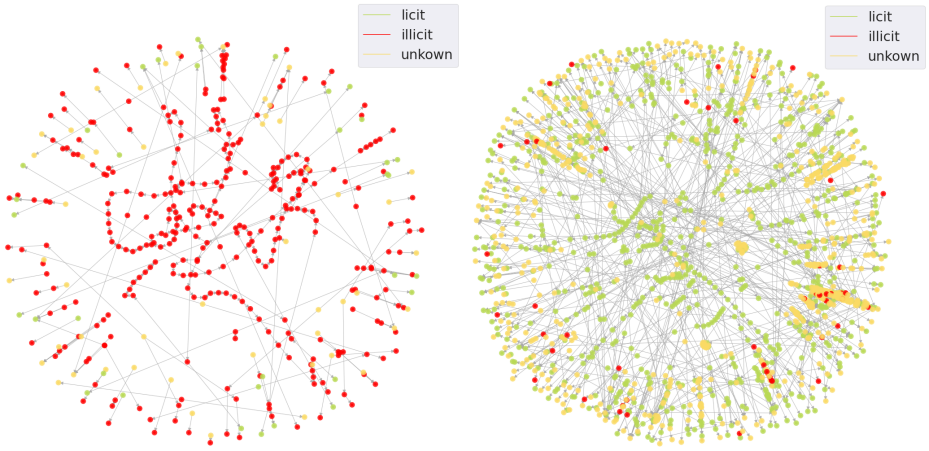


Fig. 9: Transaction graph of 1 timepoint of the AMLSim data set.

Feature Engineering For each node the total degree of in- and out-going connections over all timepoints were computed as the amount of connections might

bear valuable additional information. Furthermore, the summed up revenue as well as spending for each node over the given period of time as well as the total profit (revenue - spending) was computed. A new deposit was calculated by adding the profit to the initial one. Lastly, the transaction amount served as base for the edge weights, which was normalized by the total revenue of the source account. In total 6 additional features were created, resulting in 7 features. All computed feature values were scaled to a range between 0 and 1 (MinMaxScaler from Scikit-learn).

Elliptic data set After a first development phase the models were applied to the elliptic data set [101]. The data set and its node features are anonymized as it stems from real Bitcoin transactions. A node in the graph refers to a transaction and an edge as the Bitcoins transferred in these transactions. The data consists of 203,769 nodes and 234,355 edges of which 2% are labeled 'illicit', 21% licit, and the rest as 'unknown'. Each node has 166 features, the first 94 representing local information about the transaction, the authors name e.g. number of inputs and outputs, transaction fee, timepoint (ranging from 1-49) of ca. 2 weeks each, output volume, average amount of received or spend bitcoins and average number of incoming and outgoing transactions of the node. Due to privacy restrictions the features are not described in more details. The remaining 72 features represent aggregated information of the 1-hop neighboring transactions. In order to best simulate real-world situations only the local features were included in the analysis. Figure 10 depicts the transaction graph for 1 timepoint. There was no missing entries found in the data set.



(a) Illicit nodes. (b) Licit nodes.

Fig. 10: Transaction graph of 1 timepoint of the elliptic data set portraying (a) the illicit nodes and (b) the licit ones.

TMNL data set We also intended to apply the set of implemented models to another real-world data set of transactions from TMNL ranging over ca. 2 weeks (1 timepoint) of transactions. The data consists of millions of company bank accounts, their properties, and the transactions between them combined from ABN AMRO, ING, Rabobank, Triodos Bank and de Volksbank. Similar features as described above were generated for this data set. The data set was securely stored and could only be accessed through AWS SageMaker on the TMNL servers. For the TMNL data set the following libraries were used in addition to the once listed above: Spark [109], Pytorch [77], Spark SQL [7], and GraphFrames for graph visuzliaztions [22]. Applying the model to the real-world data set was not successful as many issues in regard to the size of the data set as well as privacy restrictions arose. In a real-world setting financial data is well protected and different python package like Pytorch Geometric, cannot just be pip installed on the servers, but first need to be reviewed and centrally added. Furthermore, the large size of the transaction graph poses additional difficulties calling for the need of environments that are able to handle big data, e.g. distributed computing and large memory.

2.6 Node Embeddings

(a) Node Embeddings with DGI + GraphSAGE for the AMLSim data set.

(b) Node Embeddings with DGI + GraphSAGE for the Elliptic data set.

Fig. 11: Node Embeddings with DGI + GraphSAGE.

A visual inspection of the node embeddings generated in the stepwise framework indicate the usefulness of supervised learning methods. After applying DGI + encoders, node2vec respectively, clusters of illicit and licit nodes seem to be

identifiable which is illustrated in Figure 11 for DGI + GraphSAGE. For all other illustrations of the node embeddings see Appendix A. The node features are illustrated in a 2D-space using Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) [65], a dimension reduction technique also suited for non-linear dimension reduction.

2.7 Evaluation procedure

The implemented algorithms were evaluated using the provided labels of the data; in case of the elliptic data set only the known lables, i.e. illicitly and licitly labeled nodes were included in the evaluation. We used the Receiver Operating Characteristic (ROC) curve [44], [80] to compare the performances of the different approaches. The ROC curves illustrate the true positive rate (TPR) against the False Positive Rate (FPR) for different thresholds between 0 and 1 (see also Table 2). The TPR is also known as sensitivity or recall and describes how well a model performs in finding the true positives out of all positives:

		Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

Table 2: Confusion Matrix showing True Positives (TP), False Positives (FP), False Negatives (FN), and True Negatives (TN).

$$TPR = \frac{TP}{TP + FN} \tag{14}$$

whereas the FPR describes 1 - specificity, i.e. the fraction of false positives a model found out of all negatives:

$$FPR = 1 - \frac{TN}{TN + FN} = \frac{FP}{TN + FP} \tag{15}$$

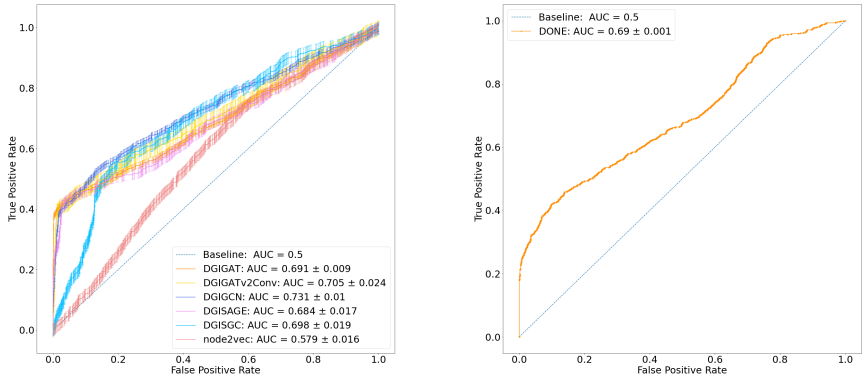
The performance of the evaluated algorithms are often quantified in a single measurement, i.e. the Area Under the Curve (AUC), the Area Under the Receiver Operating Characteristics (AUROC) respectively. A value of $AUC = 0.5$ indicates randomness, i.e. a diagonal ROC curve from the bottom left to the top right corner. On the other hand, a value of $AUC = 1$ describes optimal performance, i.e. a ROC curve going from the left bottom corner, to the left upper corner, to the right upper corner. Given the high class imbalance in the data sets, the ROC curves were additionally compared to their precision-recall curves (see also Appendix B).

3 Results

3.1 AMLSim data set

Anomaly Detection The ROC curves (see also Figure 12a) comparing DGI + encoders and node2vec to one another show that node2vec performs worse (AUC

= 57.9%) than DGI + any encoder (all AUCs $\geq$ 68.4%). The different encoders do not vary greatly between each other, but DGI + GCN seems to perform best in this case. There is no great difference between the performance of the transductive approaches, i.e. node2vec and DGI + GraphSAGE compared to the other implementations of DGI besides the lower performance of node2vec. For the joined anomaly detection algorithm, DONE (see also Figure 12b), the results show an overall AUC > 50% indicating that the illicit nodes can be identified above chance level. DONE seems to be overall sightly more instable than the stepwise approaches, but with an AUC of 69% its performance is comparable to DGI + any encoder.

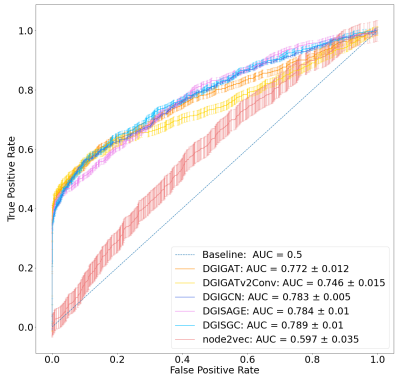


(a) ROC Curve with error bars for anomaly detection using the stepwise approach: DGI/node2vec + IF.

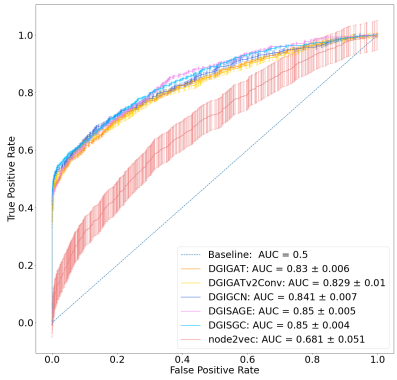
(b) ROC Curve with error bars for anomaly detection using the joined model: DONE.

Fig. 12: Evaluation of anomaly detection algorithms for the AMLSim data set.

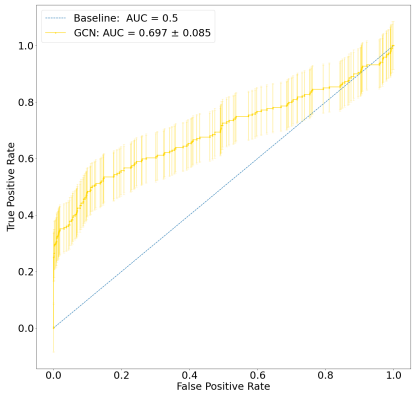
Classification In line with the results for the stepwise approach for anomaly detection, the ROC curves comparing DGI + encoders and node2vec with LR or XGBoost show that node2vec performs worse (AUC = 59.7%, AUC = 68.1%, respectively), but is also more noisy than DGI + any encoder (all AUCs $\geq$ 74.6%, all AUCs $\geq$ 82.9%, respectively) for all classification algorithms (see also Figure 13). Again, the different encoders do not vary greatly between each other. The pattern of performance of XGBoost (see also Figure 13b) is comparable to that of Logistic Regression (see also Figure 13a), however the AUCs appear to be slightly higher than those for LR. When comparing transductive versus inductive methods, similar results as for the stepwise approach for anomaly detection can be found. For the joined classification algorithm, GCN + softmax (see also Figure



(a) ROC Curve with error bars for classification using the stepwise approach: DGI/node2vec + LR.



(b) ROC Curve with error bars for classification using the stepwise approach: DGI/node2vec + XGBoost.



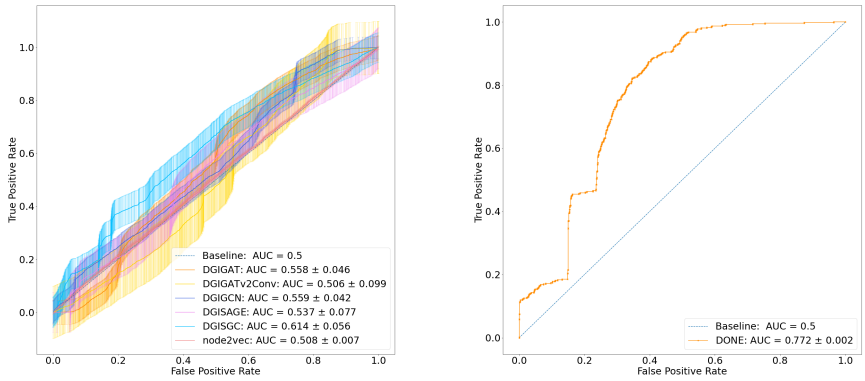
(c) ROC Curve with error bars for classification using the joined model: GCN + softmax.

Fig. 13: Evaluation of classification algorithms for the AMLSim data set.

13c), the results show an overall identification of illicit nodes above chance level with an AUC $> 50\%$. However, performance is lower when compared to any DGI + encoder for LR and XGBoost and the performance is more noisy and instable.

3.2 Elliptic data set

Anomaly Detection Unlike for AMLSim the performance (see also Figure 14a) of anomaly detection using a stepwise approach on the elliptic data set only shows correct identification of illicit nodes above chance level for some combinations of DGI + encoders. However, compared to the AMLSim data set performance is on average lower and also more noisy. The joined anomaly detection algorithm, DONE (see also Figure 14b), seems to be rather instable, although on average still above chance level and higher than for IF. For the elliptic data set only a subset of the timepoints (i.e. 26-30) could be run with DONE, which equaled just below 10k nodes. With more nodes to train the pipeline broke.

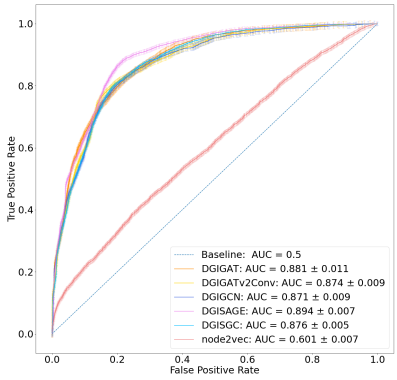


(a) ROC Curve with error bars for anomaly detection using the stepwise approach: DGI/node2vec + IF with.

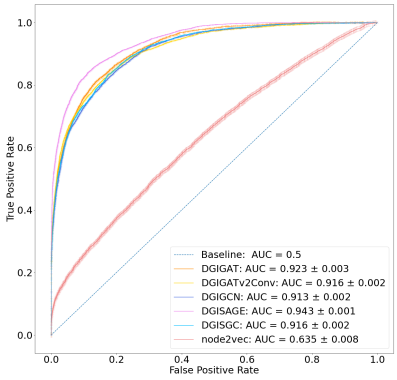
(b) ROC Curve with error bars for anomaly detection using the joined model: DONE.

Fig. 14: Evaluation of anomaly detection algorithms for the Elliptic data set.

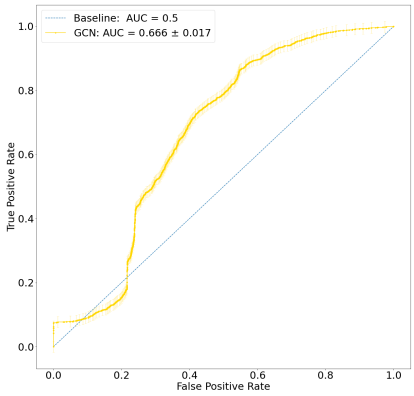
Classification The ROC curve shows for DGI + encoders high performances for all classification algorithms with all AUCs $\geq$ 87.1% (see also Figure 15). However, node2vec again is clearly outperformed by DGI + any encoder with AUCs of 60.1% for LR, and an AUC of 63.5% for XGBoost. Comparably to the AMLSim data set, the performance of DGI + the different encoders do not vary greatly between each other, however DGI + GraphSAGE seems to perform best for both stepwise approaches. Similarly to AMLSim the pattern of performance of XGBoost 15b is comparable to that of Logistic Regression, however the AUCs appear to be slightly higher than those for LR. In combination with XGBoost, DGI + GraphSAGE (AUC = 94.3%) also seems to show a slightly



(a) ROC Curve with error bars for classification using the stepwise approach: DGI/node2vec + LR.



(b) ROC Curve with error bars for classification using the stepwise approach: DGI/node2vec + XGBoost.



(c) ROC Curve with error bars for classification using the joined model: GCN + softmax.

Fig. 15: Evaluation of classification algorithms for the Elliptic data set.

higher performance than the other combinations of DGI + encoder. The joined classification algorithm, GCN + softmax, shows moderate performance with an AUC of 66.6% (see also Figure 15c), however the performance varies especially for thresholds of approximately TPR <0.4 and FPR <0.2. GCN + softmax is outperformed by the stepwise classification approaches using DGI + any decoder.

4 Discussion

The results show that in general graph-based algorithms show very promising performances in identifying illicit nodes amongst licit ones with many approaches clearly showing performances above the random baseline. For both data sets, the classification-based algorithms outperformed the anomaly detection ones for the stepwise approaches, in case of the joined approaches results are more mixed, instable, and noisy. For the stepwise approach node2vec-based implementations perform substantially worse than DGI + an encoder. This could be due to the fact that whilst node2vec performs random walks DGI learns message passing improving the performance considerably. Moreover, node2vec is only scalable to a certain extend as this method cannot generalize to new graphs nor handle dynamic ones and does not share any parameters between nodes leading to inefficient computations [110]. When generating node embeddings, the unsupervised algorithms, i.e. DGI and node2vec seem to aggregate information about each node that already preserves information that allows to differentiate between licit and illicit nodes (see also Figure 11). Over and above, node embedding algorithms + IF perform in general worse than the supervised stepwise approaches with DGI + any encoder + XGBoost performing best. Overall, the different encoders in combination with DGI perform relatively similarly, however DGI + GraphSAGE either performs best or close to best compared to the other combinations. Since it is the only transductive encoder and could also handle data of considerably larger size, the combination of DGI + GraphSAGE + XGBoost is the preferred option.

For the AMLSim data set performances of the joined approaches for anomaly detection, i.e. DONE, and classification, i.e. GCN + softmax, are comparable, for the elliptic data set DONE outperforms GCN + softmax. Furthermore, the stepwise approaches outperform the joined implementation for classification for both data sets; the anomaly detection models seem to be more noisy and instable and the differences are not comparable between stepwise and joined approaches for the AMLSim data set, for the elliptic data set DONE clearly outperforms the stepwise approaches. For the classification algorithms, the lower performance of GCN + softmax could be due to the more simplistic architecture of GCN + softmax compared to e.g. the relatively recently developed DGI approach. The GCN architecture, for example, cannot take edge direction into account compared to DGI. When comparing the performance of all approaches between the two data sets, we see higher scores for the elliptic data set compared to the AMLSim data set. The main reason presumably being a higher amount of node features in the elliptic data set which are essential to all approaches.

Our results are in line with Weber et al. (2019) [101] who also documented higher performance for supervised stepwise approaches compared to joined implementations on the elliptic data set. Furthermore, the authors also found lower performance of LR compared to a more advanced classification algorithm. As such, this research adds valuable validation to these findings and beyond that adds information about other approaches.

Lastly, the project has shown that in a real-world setting many challenges arise when working with transaction graphs. Mainly the size and the limited accessibility due to privacy protections cause issues when trying to apply these models to financial data in a real-world setting. [47].

4.1 Limitations and Outlook

Contrary to other studies, e.g. [101], we did not perform any hyperparameter tuning on the models since this would have exceeded the scope of this project as it is a time- and resource-consuming process [108]. However, results are expected to improve noticeably when doing so. Hyperparamters to include are e.g. the number of epochs, different learning rates, or the number of hidden channels as well as specific parameters for each implemented algorithm.

More importantly, however, we did not include the dynamism of transaction graphs over time in this project. Taking into account how the graph changes over time can add a further dimension and additional valuable information in countering money laundering. Transaction networks are not static, but their structure changes over time, e.g. a transaction is made to a new bank account at time point *t+1*. To capture how nodes and the relationships amongst them change over time, real-world networks can be modeled in a dynamic graph *G(t) = {V(t);E(t);Xv(t);Xe(t)}* with nodes *V*, edges *E*, attribute matrices of nodes (*Xv*) and edges (*Xe*) at time point *t* [82], [95]. Dynamic graphs include meaningful temporal signals that can give more insights into the structure of illicit transactions resulting into a higher detection rate, e.g. a node might look normal at time point *t*, but only the comparison to time point *t+1* reveals its illicit nature [62], [84]. Pareja et al. (2020) [76] for example, propose EvolveGCN on the basis of a GCN captured in a Recurrent Neural Network (RNN) structure and find higher performance of this approach compared to similar approaches. Further research is needed to integrate this approach into the investigated model. From the set of the current methods only the transductive methods are able to deal with evolving graphs, however they do not capture the dynamic nature with the current implementations.

In this project explainability of the different methods was only indirectly considered, e.g. when choosing LR as classification algorithm. Being able to infer which features or feature combinations led to identifying illicit accounts is a central challenge in successfully convicting individuals operating accounts with which money was laundering. However, the generated node embeddings are not explainable and thus even applying LR only leads to limited explainability. One such attempt is e.g. GNNExplainer [106] which tries to infer the features and structures that led a GNN to classify a node as e.g. illicit or licit. Additional research is needed to investigate how the outcome of GNNs can be interpreted and explained which serve as basis for prosecutors in a potential indictment.

We also had to limit ourselves to a set of approaches that we compared. Further research is needed to test improved joined methods that have shown to outperform their predecessors; e.g. Composition-based Multi-Relational Graph

Convolutional Networks (CompGCN) [90] showed superior results over comparable methods on a number of task including node classification. Furthermore, comparing stepwise approaches with node embeddings generated by unsupervised versus supervised methods might give valuable insights into finding the best suited approaches for analyzing financial transaction data. Lastly, the generated node embeddings could be used as input for joined methods rather than the original node features (see also [101]).

The data sets used in this project were of substantial size, however still relatively small compared to real-world transaction graphs. In course of this project we could already identify major challenges when applying these methods in a real-world setting. Additional research is needed, however to specifically investigate challenges and solutions when it comes to closing the gap between scientific research and application ensuring the developed methods can also benefit society. This can be achieved by either using a bigger simulated data set or if available, public data set, ideally however this line of research should work with real-world transaction data while ensuring privacy and security of the data .

5 Conclusion

In this project we implemented and compared different approaches with the goal to detect illicit accounts amongst a vast majority of licit ones in a financial transaction network. We implemented a variety of approaches comparing unsupervised versus supervised, stepwise versus joined approaches, and transductive versus inductive methods. In general, the graph-based methods seem to be suitable to find illicit nodes within a transaction graph, especially stepwise implementations using DGI + GraphSAGE + XGBoost show very high performances (AUCs up to 94.3% for the elliptic data set) and are the recommended technique to use. Furthermore, this project revealed valuable insights into unforeseen difficulties when applying these models to data sets in a real-world setting and thus resembles a first step into better understanding how to close the gap between research and practice.

This research contributed valuable results to those countering money laundering and outlined usable methods to find illicit accounts in a financial transaction network. However, more research is needed to expand the methods investigated in this project and their explainability in order to successfully convict individuals that use accounts to launder money.

References

[1] M. Abadi, A. Agarwal, P. Barham, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, vol. abs/1603.04467, 2016. arXiv: 1603.04467. [Online]. Available: http://arxiv.org/abs/1603.04467.

[2] A. F. Agarap, "Deep learning using rectified linear units (relu)," *CoRR*, vol. abs/1803.08375, 2018. arXiv: 1803.08375. [Online]. Available: http://arxiv.org/abs/1803.08375.

[3] C. G. Akcora, Y. Li, Y. R. Gel, and M. Kantarcioglu, "Bitcoinheist: Topological data analysis for ransomware detection on the bitcoin blockchain," *CoRR*, vol. abs/1906.07852, 2019. arXiv: 1906.07852. [Online]. Available: http://arxiv.org/abs/1906.07852.

[4] L. Akoglu, M. McGlohon, and C. Faloutsos, "Oddball: Spotting anomalies in weighted graphs," in *Advances in Knowledge Discovery and Data Mining, 14th Pacific-Asia Conference, PAKDD 2010, Hyderabad, India, June 21-24, 2010. Proceedings. Part II*, M. J. Zaki, J. X. Yu, B. Ravindran, and V. Pudi, Eds., ser. Lecture Notes in Computer Science, vol. 6119, Springer, 2010, pp. 410–421. DOI: 10.1007/978-3-642-13672-6_40. [Online]. Available: https://doi.org/10.1007/978-3-642-13672-6_40.

[5] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: A survey," *Data Min. Knowl. Discov.*, vol. 29, no. 3, pp. 626–688, 2015. DOI: 10.1007/s10618-014-0365-y. [Online]. Available: https://doi.org/10.1007/s10618-014-0365-y.

[6] W. S. Al Farizi, I. Hidayah, and M. N. Rizal, "Isolation forest based anomaly detection: A systematic literature review," in *2021 8th International Conference on Information Technology, Computer and Electrical Engineering (ICITACEE)*, IEEE, 2021, pp. 118–122.

[7] M. Armbrust, R. S. Xin, C. Lian, *et al.*, "Spark SQL: relational data processing in spark," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, T. K. Sellis, S. B. Davidson, and Z. G. Ives, Eds., ACM, 2015, pp. 1383–1394. DOI: 10.1145/2723372.2742797. [Online]. Available: https://doi.org/10.1145/2723372.2742797.

[8] S. Bandyopadhyay, L. N, S. V. Vivek, and M. N. Murty, "Outlier resistant unsupervised deep architectures for attributed network embedding," in *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3-7, 2020*, J. Caverlee, X. B. Hu, M. Lalmas, and W. Wang, Eds., ACM, 2020, pp. 25–33. DOI: 10.1145/3336191.3371788. [Online]. Available: https://doi.org/10.1145/3336191.3371788.

[9] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia,*

Canada], T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., MIT Press, 2001, pp. 585–591. [Online]. Available: https://proceedings.neurips.cc/paper/2001/hash/f106b7f99d2cb30c3db1c3cc0fde9ccb-Abstract.html.

[10] R. V. Belle, C. V. Damme, H. Tytgat, and J. D. Weerdt, "Inductive graph representation learning for fraud detection," *Expert Syst. Appl.*, vol. 193, p. 116 463, 2022. DOI: 10.1016/j.eswa.2021.116463. [Online]. Available: https://doi.org/10.1016/j.eswa.2021.116463.

[11] R. V. Belle, S. Mitrovic, and J. D. Weerdt, "Representation learning in graphs for credit card fraud detection," in *Mining Data for Financial Applications - 4th ECML PKDD Workshop, MIDAS 2019, Würzburg, Germany, September 16, 2019, Revised Selected Papers*, V. Bitetta, I. Bordino, A. Ferretti, F. Gullo, S. Pascolutti, and G. Ponti, Eds., ser. Lecture Notes in Computer Science, vol. 11985, Springer, 2019, pp. 32–46. DOI: 10.1007/978-3-030-37720-5_3. [Online]. Available: https://doi.org/10.1007/978-3-030-37720-5_3.

[12] Y. Bengio, A. C. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013. DOI: 10.1109/TPAMI.2013.50. [Online]. Available: https://doi.org/10.1109/TPAMI.2013.50.

[13] C. M. Bishop and N. M. Nasrabadi, Pattern Recognition and Machine Learning, 4. 2007, vol. 16, p. 049 901. DOI: 10.1117/1.2819119. [Online]. Available: https://doi.org/10.1117/1.2819119.

[14] E. Bisong, *Building machine learning and deep learning models on Google cloud platform: A comprehensive guide for beginners*. Apress, 2019.

[15] A. Boukerche, L. Zheng, and O. Alfandi, "Outlier detection: Methods, models, and classification," *ACM Comput. Surv.*, vol. 53, no. 3, 55:1–55:37, 2020. DOI: 10.1145/3381028. [Online]. Available: https://doi.org/10.1145/3381028.

[16] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" *CoRR*, vol. abs/2105.14491, 2021. arXiv: 2105.14491. [Online]. Available: https://arxiv.org/abs/2105.14491.

[17] M. M. Bronstein, J. Bruna, T. Cohen, and P. Velickovic, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," *CoRR*, vol. abs/2104.13478, 2021. arXiv: 2104.13478. [Online]. Available: https://arxiv.org/abs/2104.13478.

[18] I. Chami, S. Abu-El-Haija, B. Perozzi, C. Ré, and K. Murphy, "Machine learning on graphs: A model and comprehensive taxonomy," *CoRR*, vol. abs/2005.03675, 2020. arXiv: 2005.03675. [Online]. Available: https://arxiv.org/abs/2005.03675.

[19] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, 15:1–15:58, 2009. DOI: 10.1145/1541880.1541882. [Online]. Available: https://doi.org/10.1145/1541880.1541882.

[20] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," vol. abs/1603.02754, 2016. arXiv: 1603.02754. [Online]. Available: http://arxiv.org/abs/1603.02754.

[21] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, W. W. Cohen, A. McCallum, and S. T. Roweis, Eds., ser. ACM International Conference Proceeding Series, vol. 307, ACM, 2008, pp. 160–167. DOI: 10.1145/1390156.1390177. [Online]. Available: https://doi.org/10.1145/1390156.1390177.

[22] A. Dave, A. Jindal, L. E. Li, R. Xin, J. Gonzalez, and M. Zaharia, "Graphframes: An integrated API for mixing graph and relational queries," in *Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems, Redwood Shores, CA, USA, June 24 - 24, 2016*, P. A. Boncz and J. L. Larriba-Pey, Eds., ACM, 2016, p. 2. DOI: 10.1145/2960414.2960416. [Online]. Available: https://doi.org/10.1145/2960414.2960416.

[23] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *CoRR*, vol. abs/1606.09375, 2016. arXiv: 1606.09375. [Online]. Available: http://arxiv.org/abs/1606.09375.

[24] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," *CoRR*, vol. abs/2008.08692, 2020. arXiv: 2008.08692. [Online]. Available: https://arxiv.org/abs/2008.08692.

[25] U. N. O. on Drugs and Crimes, "Estimating illicit financial flows resulting from drug trafficking and other transnational organised crime: Research report," 2011.

[26] U. N. O. on Drugs and Crime. "Money laundering." (), [Online]. Available: https://www.unodc.org/unodc/en/money-laundering/overview.html. (accessed: 28.01.2022).

[27] D. Eswaran, C. Faloutsos, S. Guha, and N. Mishra, "Spotlight: Detecting anomalies in streaming graphs," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, Y. Guo and F. Farooq, Eds., ACM, 2018, pp. 1378–1386. DOI: 10.1145/3219819.3220040. [Online]. Available: https://doi.org/10.1145/3219819.3220040.

[28] Europol, *From suspicion to action, converting financial intelligence into greater operational impact*, 2017.

[29] J. Ferwerda and E. R. Kleemans, "Estimating money laundering risks: An application to business sectors in the netherlands," *European Journal on Criminal Policy and Research*, vol. 25, no. 1, pp. 45–62, 2019.

[30] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *CoRR*, vol. abs/1903.02428, 2019. arXiv: 1903.02428. [Online]. Available: http://arxiv.org/abs/1903.02428.

[31] J. R. Firth, "A synopsis of linguistic theory, 1930-1955," *Studies in linguistic analysis*, 1957.

[32] S. Fortunato, "Community detection in graphs," *CoRR*, vol. abs/0906.0612, 2009. arXiv: 0906.0612. [Online]. Available: http://arxiv.org/abs/0906.0612.

[33] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *CoRR*, vol. abs/1705.02801, 2017. arXiv: 1705.02801. [Online]. Available: http://arxiv.org/abs/1705.02801.

[34] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," vol. abs/1607.00653, 2016. arXiv: 1607.00653. [Online]. Available: http://arxiv.org/abs/1607.00653.

[35] F. E. Grubbs, "Procedures for detecting outlying observations in samples," *Technometrics*, vol. 11, no. 1, pp. 1–21, 1969.

[36] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[37] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *CoRR*, vol. abs/1706.02216, 2017. arXiv: 1706.02216. [Online]. Available: http://arxiv.org/abs/1706.02216.

[38] ——, "Representation learning on graphs: Methods and applications," *IEEE Data Eng. Bull.*, vol. 40, no. 3, pp. 52–74, 2017. [Online]. Available: http://sites.computer.org/debull/A17sept/p52.pdf.

[39] C. R. Harris, K. J. Millman, S. van der Walt, *et al.*, "Array programming with numpy," *CoRR*, vol. abs/2006.10256, 2020. arXiv: 2006.10256. [Online]. Available: https://arxiv.org/abs/2006.10256.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," vol. abs/1502.01852, 2015. arXiv: 1502.01852. [Online]. Available: http://arxiv.org/abs/1502.01852.

[41] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart, "Distributed representations," in *The Philosophy of Artificial Intelligence*, ser. Oxford readings in philosophy, M. A. Boden, Ed., Oxford University Press, 1990, pp. 248–280.

[42] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," *CoRR*, vol. abs/1808.06670, 2018. arXiv: 1808.06670. [Online]. Available: http://arxiv.org/abs/1808.06670.

[43] P. D. Hoff, A. E. Raftery, and M. S. Handcock, "Latent space approaches to social network analysis," *Journal of the american Statistical association*, vol. 97, no. 460, pp. 1090–1098, 2002.

[44] M. Hoogendoorn and B. Funk, "Machine learning for the quantified self - on the art of learning from sensory data," Cognitive Systems Monographs,

vol. 35, 2018. DOI: 10.1007/978-3-319-66308-1. [Online]. Available: https://doi.org/10.1007/978-3-319-66308-1.

[45] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, 2007. DOI: 10.1109/MCSE.2007.55. [Online]. Available: https://doi.org/10.1109/MCSE.2007.55.

[46] F. Jiang, L. Zheng, J. Xu, and P. S. Yu, "FI-GRL: fast inductive graph representation learning via projection-cost preservation," *CoRR*, vol. abs/1809.08079, 2018. arXiv: 1809.08079. [Online]. Available: http://arxiv.org/abs/1809.08079.

[47] S. H. Kaisler, F. Armour, J. A. Espinosa, and W. H. Money, "Big data: Issues and challenges moving forward," in *46th Hawaii International Conference on System Sciences, HICSS 2013, Wailea, HI, USA, January 7-10, 2013*, IEEE Computer Society, 2013, pp. 995–1004. DOI: 10.1109/HICSS.2013.645. [Online]. Available: https://doi.org/10.1109/HICSS.2013.645.

[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980.

[49] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *CoRR*, vol. abs/1609.02907, 2016. arXiv: 1609.02907. [Online]. Available: http://arxiv.org/abs/1609.02907.

[50] ——, "Variational graph auto-encoders," *CoRR*, vol. abs/1611.07308, 2016. arXiv: 1611.07308. [Online]. Available: http://arxiv.org/abs/1611.07308.

[51] L. A. Kurgan and P. Musílek, "A survey of knowledge discovery and data mining process models," *Knowl. Eng. Rev.*, vol. 21, no. 1, pp. 1–24, 2006. DOI: 10.1017/S0269888906000737. [Online]. Available: https://doi.org/10.1017/S0269888906000737.

[52] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nat.*, vol. 521, no. 7553, pp. 436–444, 2015. DOI: 10.1038/nature14539. [Online]. Available: https://doi.org/10.1038/nature14539.

[53] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[54] J. Li, H. Dani, X. Hu, and H. Liu, "Radar: Residual analysis for anomaly detection in attributed networks," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, C. Sierra, Ed., ijcai.org, 2017, pp. 2152–2158. DOI: 10.24963/ijcai.2017/299. [Online]. Available: https://doi.org/10.24963/ijcai.2017/299.

[55] C. Liang, Z. Liu, B. Liu, *et al.*, "Uncovering insurance fraud conspiracy with network learning," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, B. Piwowarski, M. Chevalier, É. Gaussier, Y. Maarek, J. Nie, and F. Scholer, Eds., ACM, 2019,

pp. 1181–1184. DOI: 10.1145/3331184.3331372. [Online]. Available: https://doi.org/10.1145/3331184.3331372.

[56] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation forest," in *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, IEEE Computer Society, 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17. [Online]. Available: https://doi.org/10.1109/ICDM.2008.17.

[57] ——, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discov. Data*, vol. 6, no. 1, 3:1–3:39, 2012. DOI: 10.1145/2133360.2133363. [Online]. Available: https://doi.org/10.1145/2133360.2133363.

[58] K. Liu, Y. Dou, Y. Zhao, *et al.*, "Benchmarking node outlier detection on graphs," *CoRR*, vol. abs/2206.10071, 2022. DOI: 10.48550/arXiv.2206.10071. arXiv: 2206.10071. [Online]. Available: https://doi.org/10.48550/arXiv.2206.10071.

[59] Z. Liu, C. Chen, X. Yang, J. Zhou, X. Li, and L. Song, "Heterogeneous graph neural networks for malicious account detection," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, A. Cuzzocrea, J. Allan, N. W. Paton, *et al.*, Eds., ACM, 2018, pp. 2077–2085. DOI: 10.1145/3269206.3272010. [Online]. Available: https://doi.org/10.1145/3269206.3272010.

[60] E. Lopez-Rojas, A. Elmir, and S. Axelsson, "Paysim: A financial mobile money simulator for fraud detection," in *28th European Modeling and Simulation Symposium, EMSS, Larnaca*, Dime University of Genoa, 2016, pp. 249–255.

[61] L. Lv, J. Cheng, N. Peng, M. Fan, D. Zhao, and J. Zhang, "Auto-encoder based graph convolutional networks for online financial anti-fraud," in *IEEE Conference on Computational Intelligence for Financial Engineering & Economics, CIFEr 2019, Shenzhen, China, May 4-5, 2019*, IEEE, 2019, pp. 1–6. DOI: 10.1109/CIFEr.2019.8759109. [Online]. Available: https://doi.org/10.1109/CIFEr.2019.8759109.

[62] X. Ma, J. Wu, S. Xue, J. Yang, Q. Z. Sheng, and H. Xiong, "A comprehensive survey on graph anomaly detection with deep learning," *CoRR*, vol. abs/2106.07178, 2021. arXiv: 2106.07178. [Online]. Available: https://arxiv.org/abs/2106.07178.

[63] A. L. Maas, A. Y. Hannun, A. Y. Ng, *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, Citeseer, vol. 30, 2013, p. 3.

[64] M. W. Mahoney and P. Drineas, "CUR matrix decompositions for improved data analysis," *Proc. Natl. Acad. Sci. USA*, vol. 106, no. 3, pp. 697–702, 2009. DOI: 10.1073/pnas.0803205106. [Online]. Available: https://doi.org/10.1073/pnas.0803205106.

[65] L. McInnes and J. Healy, "UMAP: uniform manifold approximation and projection for dimension reduction," *CoRR*, vol. abs/1802.03426, 2018.

arXiv: 1802.03426. [Online]. Available: http://arxiv.org/abs/1802.03426.

[66] W. McKinney *et al.*, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, Austin, TX, vol. 445, 2010, pp. 51–56.

[67] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2013. [Online]. Available: http://arxiv.org/abs/1301.3781.

[68] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *CoRR*, vol. abs/1310.4546, 2013. arXiv: 1310.4546. [Online]. Available: http://arxiv.org/abs/1310.4546.

[69] S. Mitrovic, B. Baesens, W. Lemahieu, and J. D. Weerdt, "Tcc2vec: Rfm-informed representation learning on call graphs for churn prediction," *Inf. Sci.*, vol. 557, pp. 270–285, 2021. DOI: 10.1016/j.ins.2019.02.044. [Online]. Available: https://doi.org/10.1016/j.ins.2019.02.044.

[70] A. Mnih and G. E. Hinton, "A scalable hierarchical distributed language model," D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., pp. 1081–1088, 2008. [Online]. Available: https://proceedings.neurips.cc/paper/2008/hash/1e056d2b0ebd5c878c550da6ac5d3724-Abstract.html.

[71] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, J. Fürnkranz and T. Joachims, Eds., Omnipress, 2010, pp. 807–814. [Online]. Available: https://icml.cc/Conferences/2010/papers/432.pdf.

[72] U. Nations, "United nations convention against illicit traffic in narcotic drugs and psychotropic substances, 1988: Adopted by the conference at its sixth plenary meeting on 19 december 1988," 1988.

[73] T. E. Oliphant, *A guide to NumPy*. Trelgol Publishing USA, 2006, vol. 1.

[74] G. Pang, L. Cao, L. Chen, and H. Liu, "Learning representations of ultrahigh-dimensional data for random distance-based outlier detection," *CoRR*, vol. abs/1806. 04808, 2018. arXiv: 1806.04808. [Online]. Available: http://arxiv.org/abs/1806.04808.

[75] G. Pang, C. Shen, L. Cao, and A. van den Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surv.*, vol. 54, no. 2, 38:1–38:38, 2021. DOI: 10.1145/3439950. [Online]. Available: https://doi.org/10.1145/3439950.

[76] A. Pareja, G. Domeniconi, J. Chen, *et al.*, "Evolvegcn: Evolving graph convolutional networks for dynamic graphs," vol. abs/1902.10191, 2019. arXiv: 1902.10191. [Online]. Available: http://arxiv.org/abs/1902.10191.

[77] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *CoRR*, vol. abs/1912.01703, 2019. arXiv: 1912.01703. [Online]. Available: http://arxiv.org/abs/1912.01703.

[78] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[79] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," *CoRR*, vol. abs/1403.6652, 2014. arXiv: 1403.6652. [Online]. Available: http://arxiv.org/abs/1403.6652.

[80] W. W. Peterson, T. G. Birdsall, and W. C. Fox, "The theory of signal detectability," *Trans. IRE Prof. Group Inf. Theory*, vol. 4, pp. 171–212, 1954. DOI: 10.1109/TIT.1954.1057460. [Online]. Available: https://doi.org/10.1109/TIT.1954.1057460.

[81] T. Pourhabibi, K. Ong, B. Kam, and Y. L. Boo, "Fraud detection: A systematic literature review of graph-based anomaly detection approaches," *Decis. Support Syst.*, vol. 133, p. 113 303, 2020. DOI: 10.1016/j.dss.2020.113303. [Online]. Available: https://doi.org/10.1016/j.dss.2020.113303.

[82] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, and N. F. Samatova, "Anomaly detection in dynamic networks: A survey," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 7, no. 3, pp. 223–247, 2015.

[83] S. X. Rao, S. Zhang, Z. Han, *et al.*, "Xfraud: Explainable fraud transaction detection on heterogeneous graphs," *CoRR*, vol. abs/2011.12193, 2020. arXiv: 2011.12193. [Online]. Available: https://arxiv.org/abs/2011.12193.

[84] R. A. Rossi, B. Gallagher, J. Neville, and K. Henderson, "Modeling dynamic behavior in large evolving graphs," in *Sixth ACM International Conference on Web Search and Data Mining, WSDM 2013, Rome, Italy, February 4-8, 2013*, S. Leonardi, A. Panconesi, P. Ferragina, and A. Gionis, Eds., ACM, 2013, pp. 667–676. DOI: 10.1145/2433396.2433479. [Online]. Available: https://doi.org/10.1145/2433396.2433479.

[85] J. Sachs, C. Kroll, G. Lafortune, G. Fuller, and F. Woelm, *Sustainable Development Report 2021*. Cambridge University Press, 2021. DOI: 10.1017/9781009106559.

[86] S. Thudumu, P. Branch, J. Jin, and J. J. Singh, "A comprehensive survey of anomaly detection techniques for high dimensional big data," *J. Big Data*, vol. 7, no. 1, p. 42, 2020. DOI: 10.1186/s40537-020-00320-x. [Online]. Available: https://doi.org/10.1186/s40537-020-00320-x.

[87] "Transactie monitoring nederland (tmnl)." (), [Online]. Available: https://tmnl.nl/. (accessed: 28.01.2022).

[88] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecu-

rity data streams," *CoRR*, vol. abs/1710. 00811, 2017. arXiv: 1710.00811. [Online]. Available: http://arxiv.org/abs/1710.00811.

[89] B. Unger, J. Ferwerda, I. Koetsier, *et al.*, *Aard en omvang van criminele bestedingen*, 2018.

[90] S. Vashishth, S. Sanyal, V. Nitin, and P. P. Talukdar, "Composition-based multi-relational graph convolutional networks," *CoRR*, vol. abs/1911.03082, 2019. arXiv: 1911.03082. [Online]. Available: http://arxiv.org/abs/1911.03082.

[91] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *CoRR*, vol. abs/1706. 03762, 2017. arXiv: 1706.03762. [Online]. Available: http://arxiv.org/abs/1706.03762.

[92] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *CoRR*, vol. abs/1710. 10903, 2017. arXiv: 1710.10903. [Online]. Available: http://arxiv.org/abs/1710.10903.

[93] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," *CoRR*, vol. abs/1809.10341, 2018. arXiv: 1809.10341. [Online]. Available: http://arxiv.org/abs/1809.10341.

[94] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, 2010. DOI: 10.5555/1756006.1953039. [Online]. Available: https://dl.acm.org/doi/10.5555/1756006.1953039.

[95] A. Z. Wang, R. Ying, P. Li, N. Rao, K. Subbian, and J. Leskovec, "Bipartite dynamic representations for abuse detection," in *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, F. Zhu, B. C. Ooi, and C. Miao, Eds., ACM, 2021, pp. 3638–3648. DOI: 10.1145/3447548.3467141. [Online]. Available: https://doi.org/10.1145/3447548.3467141.

[96] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, and R. Rastogi, Eds., ACM, 2016, pp. 1225–1234. DOI: 10.1145/2939672.2939753. [Online]. Available: https://doi.org/10.1145/2939672.2939753.

[97] D. Wang, J. Lin, P. Cui, *et al.*, "A semi-supervised graph attentive network for financial fraud detection," *CoRR*, vol. abs/2003.01171, 2020. arXiv: 2003.01171. [Online]. Available: https://arxiv.org/abs/2003.01171.

[98] J. Wang, S. Zhang, Y. Xiao, and R. Song, "A review on graph neural network methods in financial applications," *CoRR*, vol. abs/2111.15367, 2021. arXiv: 2111.15367. [Online]. Available: https://arxiv.org/abs/2111.15367.

[99] Z. Wang and C. Lan, "Towards a hierarchical bayesian model of multi-view anomaly detection," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, C. Bessiere, Ed., ijcai.org, 2020, pp. 2420–2426. DOI: 10.24963/ijcai.2020/335. [Online]. Available: https://doi.org/10.24963/ijcai.2020/335.

[100] M. Weber, J. Chen, T. Suzumura, *et al.*, "Scalable graph learning for anti-money laundering: A first look," *CoRR*, vol. abs/1812.00076, 2018. arXiv: 1812.00076. [Online]. Available: http://arxiv.org/abs/1812.00076.

[101] M. Weber, G. Domeniconi, J. Chen, *et al.*, "Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics," *CoRR*, vol. abs/1908.02591, 2019. arXiv: 1908.02591. [Online]. Available: http://arxiv.org/abs/1908.02591.

[102] D. B. West *et al.*, *Introduction to graph theory*. Prentice hall Upper Saddle River, 2001, vol. 2.

[103] R. Wirth and J. Hipp, "Crisp-dm: Towards a standard process model for data mining," in *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, Manchester, vol. 1, 2000, pp. 29–40.

[104] F. Wu, T. Zhang, A. H. S. Jr., C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," vol. abs/ 1902.07153, 2019. arXiv: 1902.07153. [Online]. Available: http://arxiv.org/abs/1902.07153.

[105] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 32, no. 1, pp. 4–24, 2021. DOI: 10.1109/TNNLS.2020.2978386. [Online]. Available: https://doi.org/10.1109/TNNLS.2020.2978386.

[106] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 9240–9251. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/d80b7040b773199015de6d3b4293c8ff-Abstract.html.

[107] D. C. C. Yip and M. van Dijck Nemcsik, *Anti-money laundering transaction monitoring systems implementation : finding anomalies*, ser. Wiley and SAS business series. John Wiley & Sons, 13, ISBN: 9781119381808.

[108] T. Yu and H. Zhu, "Hyper-parameter optimization: A review of algorithms and applications," *CoRR*, vol. abs/2003.05689, 2020. arXiv: 2003.05689. [Online]. Available: https://arxiv.org/abs/2003.05689.

[109] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud'10, Boston, MA, USA, June 22, 2010*, E. M. Nahum and D. Xu, Eds., USENIX Association, 2010.

[Online]. Available: https://www.usenix.org/conference/hotcloud-10/spark-cluster-computing-working-sets.

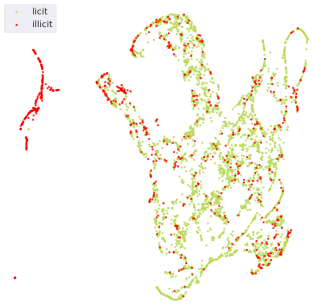
[110] J. Zhou, G. Cui, S. Hu, *et al.*, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020. DOI: 10.1016/j.aiopen.2021.01.001. [Online]. Available: https://doi.org/10.1016/j.aiopen.2021.01.001.

Appendix

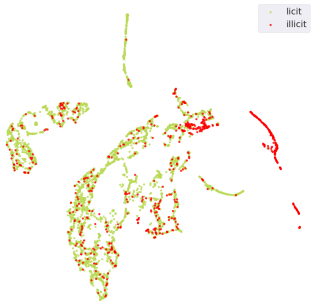
Appendix A: Node Embeddings

The node embeddings were created using DGI + an encoder as well as node2vec.

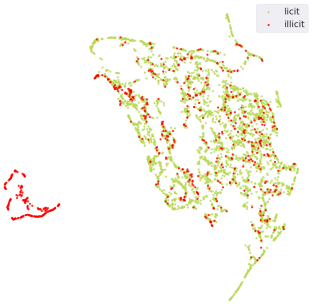
AMLSim data set Node Embeddings with DGI + encoder.



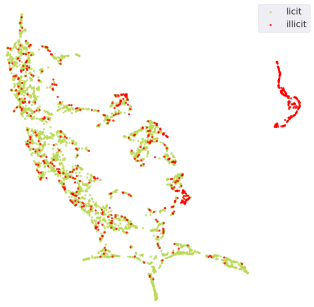
(a) Node Embeddings with DGI + GCN for the AMLSim data set.



(b) Node Embeddings with DGI + SGCN for the AMLSim data set.



(c) Node Embeddings with DGI + GAT for the AMLSim data set.



(d) Node Embeddings with DGI + GATv2Conv for the AMLSim data set.

Fig. 16: Node Embeddings with DGI + encoder for the AMLSim data set.

Node Embeddings with node2vec.

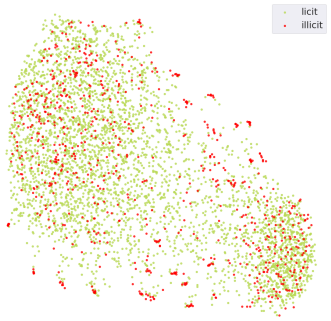
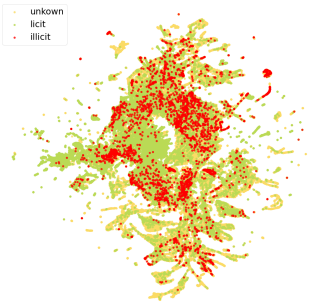


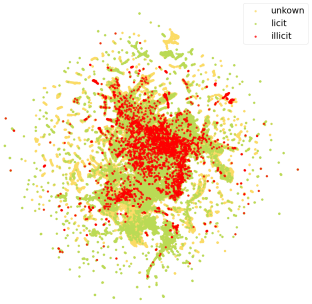
Fig. 17: Node Embeddings with node2vec for the AMLSim data set.

Elliptic data set Node Embeddings with DGI + encoder.

(a) Node Embeddings with DGI + GCN for the Elliptic data set.



(b) Node Embeddings with DGI + SGCN for the Elliptic data set.



(c) Node Embeddings with DGI + GAT for the Elliptic data set.

(d) Node Embeddings with DGI + GATv2Conv for the Elliptic data set.

Fig. 18: Node Embeddings with DGI + encoder for the Elliptic data set.

Node Embeddings with node2vec.

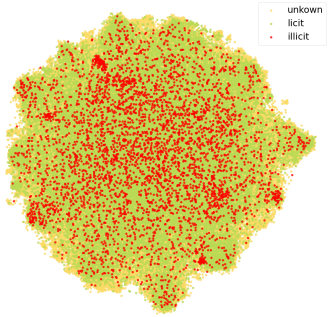
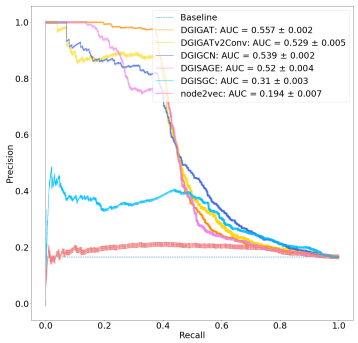


Fig. 19: Node Embeddings with node2vec for the Elliptic data set.

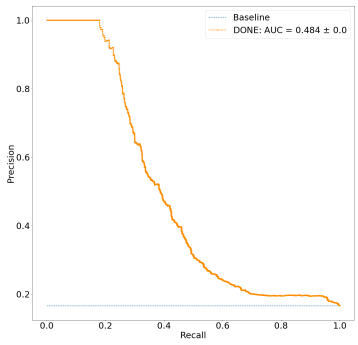
Appendix B: Precision-Recall curves

Besides the ROC curves, the performance of the different approaches was compared to one another in terms of the trade-off between precision and recall, i.e. precision-recall curves presented below.

AMLSim data set - Anomaly Detection Precision-recall curves for anomaly detection of the AMLSim data set.



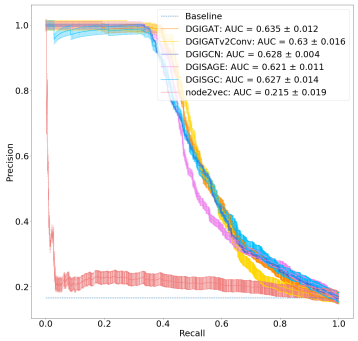
(a) Precision-Recall Curve with error bars for anomaly detection using the stepwise approach: DGI/node2vec + IF.



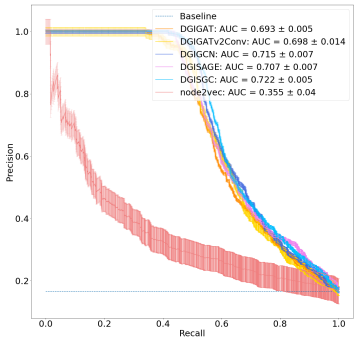
(b) Precision-Recall Curve with error bars for anomaly detection using the joined model: DONE.

Fig. 20: Evaluation of anomaly detection for the AMLSim data set: trade-off between precision and recall.

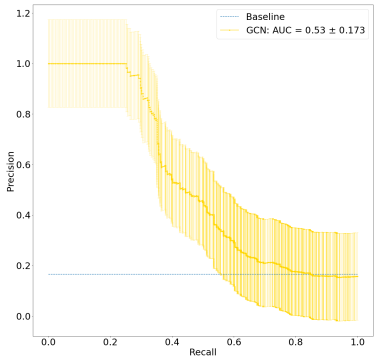
AMLSim data set - Classification Precision-recall curves for classification of the AMLSim data set.



(a) Precision-Recall Curve with error bars for classification using the stepwise approach: DGI/node2vec + LR.



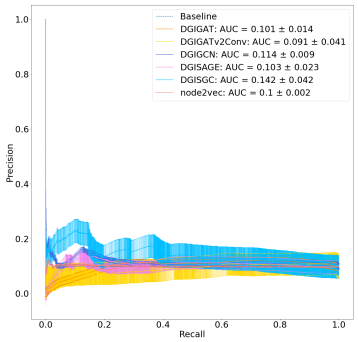
(b) Precision-Recall Curve with error bars for classification using the stepwise approach: DGI/node2vec + XGBoost.



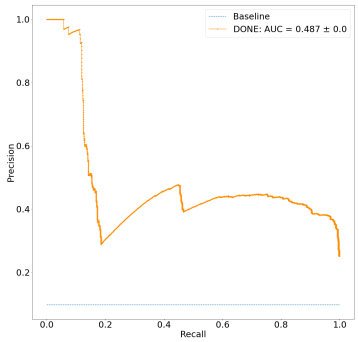
(c) Precision-Recall Curve with error bars for classification using the joined model: GCN + softmax.

Fig. 21: Evaluation of classification for the AMLSim data set: trade-off between precision and recall.

Elliptic data set - Anomaly Detection Precision-recall curves for anomaly detection of the Elliptic data set.



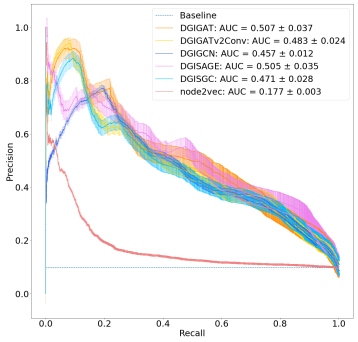
(a) Precision-Recall Curve with error bars for anomaly detection using the stepwise approach: DGI/node2vec + IF with.



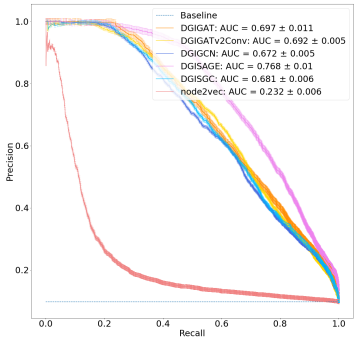
(b) Precision-Recall Curve with error bars for anomaly detection using the joined model: DONE.

Fig. 22: Evaluation of anomaly detection for the Elliptic data set: trade-off between precision and recall.

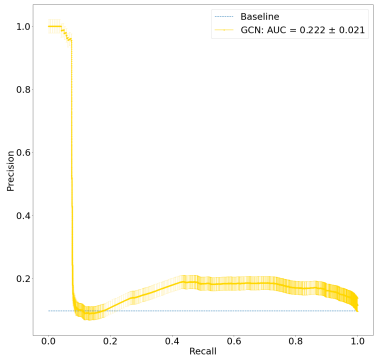
Elliptic data set - Classification Precision-recall curves for classification of the Elliptic data set.



(a) Precision-Recall Curve with error bars for classification using the stepwise approach: DGI/node2vec + LR.



(b) Precision-Recall Curve with error bars for classification using the stepwise approach: DGI/node2vec + XGBoost.



(c) Precision-Recall Curve with error bars for classification using the joined model: GCN + softmax.

Fig. 23: Evaluation of classification for the Elliptic data set: trade-off between precision and recall.