

Inspection-L: Self-Supervised GNN Node Embeddings for Money Laundering Detection in Bitcoin

Wai Weng Lo^{a,*}, Gayan K. Kulatilleke^a, Mohanad Sarhan^a, Siamak Layeghy^a, Marius Portmann^a

^aThe University of Queensland, Brisbane, Australia

Abstract

Criminals have become increasingly experienced in using cryptocurrencies, such as Bitcoin, for money laundering. The use of cryptocurrencies can hide criminal identities and transfer hundreds of millions of dollars of dirty funds through their criminal digital wallets. However, this is considered a paradox because cryptocurrencies are goldmines for open-source intelligence, giving law enforcement agencies more power when conducting forensic analyses. This paper proposed Inspection-L, a graph neural network (GNN) framework based on a self-supervised Deep Graph Infomax (DGI) and Graph Isomorphism Network (GIN), with supervised learning algorithms, namely Random Forest (RF), to detect illicit transactions for anti-money laundering (AML). To the best of our knowledge, our proposal is the first to apply self-supervised GNNs to the problem of AML in Bitcoin. The proposed method was evaluated on the Elliptic dataset and shows that our approach outperforms the state-of-the-art in terms of key classification metrics, which demonstrates the potential of self-supervised GNN in the detection of illicit cryptocurrency transactions.

Keywords: graph neural networks; machine learning; forensics; anomaly detection; cryptocurrencies

1. Introduction

The advent of the first cryptocurrency—Bitcoin [1]—has revolutionized the conventional financial ecosystem, as it enables low-cost, near-anonymous, peer-to-peer cash transfers within and across various borders. Due to its pseudonymity, many cybercriminals, terrorists, and hackers have started to use cryptocurrency for illegal transactions. For example, the WannaCry ransomware attack used Bitcoin [2] as the payment method due to its non-traceability. The criminals received nearly 3.4 million (46.4 BTC) within four days of the WannaCry attack [2]. Therefore, effective detection of illicit transactions in Bitcoin transaction graphs is essential for preventing illegal transactions. Paradoxically, cryptocurrencies are goldmines for open-source intelligence, as transaction network data are publicly available, enabling law enforcement agencies to conduct a forensic analysis of the transaction's linkages and flows. However, the problem is challenging for law enforcement agencies, owing to its volume¹, the untraceable p2p cross-border nature of Bitcoin transactions, and the use of technologies such as mixers and tumblers.

Graph representation learning has shown great potential for detecting money laundering activities using cryptocurrencies. GNNs are tailored to applications with graph-structured data, such as the social sciences, chemistry, and telecommunications,

and can leverage the inherent structure of the graph data by building relational inductive biases into the deep learning architecture. This provides the ability to learn, reason, and generalize from the graph data, inspired by the concept of message propagation [3].

The Bitcoin transaction flow data can naturally be represented in graph format. A graph is constructed from the raw Bitcoin data and labeled such that the nodes represent transactions and the edges represent the flow of Bitcoin currency (BTC) from one transaction to the next in the adjacency matrix. Both the topological information and the information contained in the node features are crucial for detecting illicit transactions.

This paper proposes Inspection-L, a Graph Neural Network (GNN) framework based on an enhanced self-supervised Deep Graph Infomax (DGI) [4] and supervised Random Forest (RF)-based classifier to detect illicit transactions for AML.

Specifically, we investigate the Elliptic dataset [5], a realistic, partially labeled Bitcoin temporal graph-based transaction dataset consisting of real entities belonging to licit (e.g., wallet, miners), illicit entities (e.g., scams, terrorist organizations, ransomware), and unknown transaction categories. The proposed Inspection-L framework aims to detect illegal transactions based on graph representation learning in a self-supervised manner. Current graph machine learning approaches, such as [5], generally apply supervised graph neural network approaches to the detection of illicit transactions. However, supervised learning requires manual labeling. In the AML scenario, building an effective model that utilizes unknown label data is required, since human's labeling Bitcoin data could be costly and ineffective. It also only performs well when the labels are enough. Thus, exploiting unlabeled data to improve performance is crit-

*Corresponding author

Email addresses: w.w.lo@uq.net.au (Wai Weng Lo), g.kulatilleke@uq.net.au (Gayan K. Kulatilleke), m.sarhan@uq.net.au (Mohanad Sarhan), siamak.layeghy@uq.net.au (Siamak Layeghy), marius@itee.uq.edu.au (Marius Portmann)

¹As of 2022 Aug 09, the volume of the entire BTC transaction record, the blockchain, is 420GB, with an average growth rate of 129%

ical for AML. On the other hand, self-supervised graph neural network algorithms [6][7] allow for the unknown label data to be exploited, which can improve the quality of representation for the downstream tasks such as fraud transaction detection in Bitcoin. Furthermore, in supervised learning, GNN is limited to capturing K-hop neighbor information; for example, once the hops of the neighbor are larger than k, the supervised learning GNN fails to capture that node information.

In this paper, we applied DGI self-supervised learning to capture the global graph information, as this is not limited to capturing the K-layer neighborhood information, where every node can access the entire graph’s structural pattern and node information using random shuffle node features. The DGI discriminator tries to determine wherever the node feature is shuffled or not. Thus, every node can access global parts of the node’s properties, rather than K-layer neighborhood information.

We demonstrate how the self-supervised DGI algorithm can be integrated with standard machine learning classification algorithms, i.e., Random Forest, to build an efficient anti-money-laundering detection system. We show that our Inspection-L method outperforms the state-of-the-art in terms of F1 score.

In summary, the key contributions of this paper are:

- Different from most existing works, which typically use supervised graph representation learning to generate node embeddings for illegal transaction detection, we use a self-supervised learning approach to learn the node embeddings without using any labels.
- The proposed Inspection-L is based on a self-supervised DGI combined with the Random Forest (RF) supervised machine learning algorithms, to capture topological information and node features in the transaction graph to detect illegal transactions. To the best of our knowledge, our proposal is the first to utilize self-supervised GNNs to generate node embeddings for AML in Bitcoin.
- The comprehensive evaluation of the proposed framework using the Elliptic benchmark datasets demonstrates superior performance compared to other, supervised machine learning approaches.

2. RELATED WORKS

Mark et al. [5] created and published the Elliptic dataset, a temporal graph-based Bitcoin transaction dataset consisting of over 200K Bitcoin node transactions, 234K payment edges, and 49 transaction graphs with distinct time steps. Each of the transaction nodes was labeled as a "licit", "illicit", or "unknown" entity. They evaluated the Elliptic dataset using various machine learning methods, including Logistic Regression (LR), Random Forest (RF), Multilayer Perceptrons (MLP) [8], Graph Convolutional Networks (GCNs) [9] and EvolveGCN [10]. They retrieved a recall score in the illicit category of 0.67 using RF and 0.51 using GCNs.

Yining et al. [11] collected the Bitcoin transaction graph data between July 2014 and May 2017 by running a Bitcoin client and used an external trusted source, "Wallet Explorer", a website that tracks Bitcoin wallets, to label the data. They first highlighted the differences between money laundering and regular transactions using network centrality such as PageRank, clustering coefficient [12], then used a node2vec-based [13] classifier to classify money laundering transactions. The research also indicated that statistical information, such as in-degree/out-degree, number of weakly connected components, and sum/mean/standard deviation of the output values, could distinguish money laundering transactions from legal transactions. However, this approach only considers graph topological patterns, without considering node features. Vassallo et al. [14] focused on the detection of illicit cryptocurrency activities (e.g., scams, terrorism financing, and Ponzi schemes). Their proposed detection framework is based on Adaptive Stacked eXtreme Gradient Boosting (ASXGB), an enhanced variation of eXtreme Gradient Boosting (XGBoost). ASXGB was evaluated using the Elliptic dataset, and the results demonstrate its superiority at both the account and transaction levels.

Chaehyeon et al. [15] applied supervised machine learning algorithms to classify illicit nodes in the Bitcoin network. They used two supervised machine learning models, namely, Random Forest (RF) and Artificial Neural Network (ANN) [8] to detect illegal transactions. First, they collected the legal and illegal Bitcoin data from the forum sites "Wallet Explorer" and "Blockchain Explorer". Next, they performed feature extraction based on the characteristics of Bitcoin transactions, such as transaction fees and transaction size. The extracted features were labeled legal or illegal for supervised training. The results indicated that relatively high F1 scores could be achieved; specifically, ANN and RF achieved 0.89 and 0.98 F1 scores, respectively. In [16] proposed using GCNs intertwined with linear layers to classify illicit nodes of the Elliptic dataset [5]. An overall classification accuracy and recall of 97.40% and 0.67, respectively, can be achieved to detect illicit transactions. In [17], the authors used an autoencoder with graph embedding to detect mixing and demixing services for Bitcoin cryptocurrency. They first applied graph node embedding to generate the node representation; then, a K-means algorithm was applied to cluster the node embeddings to detect mixing and demixing services. The proposed model was evaluated based on real-world Bitcoin datasets to evaluate the model’s effectiveness, and the results demonstrate that the proposed model can effectively perform demix/mixing service anomaly detection.

Lorenz et al. [18] proposed active learning techniques by using a minimum number of labels to achieve a high rate of detection of illicit transactions on the Elliptic dataset. In [19], the authors applied unsupervised learning to detect suspicious nodes in the Bitcoin transaction graph. They used various kinds of unsupervised machine learning algorithms, such as K-means and Gaussian Mixture models, to cluster normal and illicit nodes. However, since the Bitcoin transaction dataset they used lacked ground-truth labels, they simply used the internal index to validate the clustering algorithm, without confirming that those nodes are actually malicious transactions. Monamo et al. [20]

applied trimmed-Kmeans to detect fraud in the Bitcoin network. They used various graph centrality measures (i.e. in degree, out-degree of the Bitcoin transactions) and currency features (i.e. the total amount sent), which were then used for Bitcoin transaction clustering. However, similar to [19], due to the unavailability of ground-truth labels, they used clustering performance metrics such as "within the sum of squares", without being able to validate the true nature of the Bitcoin transaction anomalies. Shucheng et al. [21] proposed SIEGE, a self-supervised graph learning approach for Ethereum phishing scam detection, using two pretext tasks to generate node embeddings without using labels and an incremental paradigm to capture data distribution changes for over half a year. However, a significant limitation of this approach is that it does not consider the Bitcoin context and is limited to detecting Ethereum phishing scams. Additionally, their simple application of GCNs[9] in the pretext task phase is much less effective than the Weisfeiler–Lehman (1-WL) test[22].

In contrast with related studies, our approach can detect not only phishing scams but also other illicit transactions, such as terrorist organizations, ransomware and Ponzi schemes, by utilizing the Elliptic dataset [5].

3. BACKGROUND

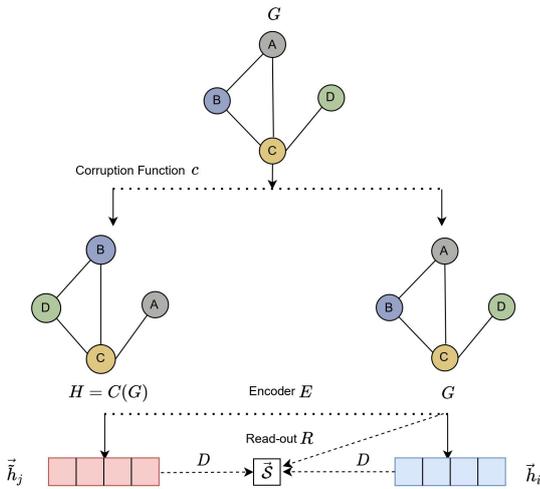


Figure 1: Overview of Deep Graph Infomax

The main innovation of our proposed model is its use of DGI [4] with our proposed GIN encoder to learn node embeddings in a self-supervised manner. Then, the node embeddings can be treated as enhanced features and be combined with the raw features for standard supervised RF machine learning algorithms to classify illicit transaction. This has a clear advantage over simple features, as inputs to overall graph-structured patterns are available for the downstream classifier.

Consequently, the current graph-based approaches [5] [16] try to apply a supervised GCN-based approach to capture the overall graph-structured patterns. However, the main limitation is that GCN can only capture the neighborhood information of

limited K layers, not the global view graph and node information, due to the threat of overfitting. While some models, such as FDGATII [23], are capable of a larger K, these are still limited by their layer structure and finite k. On the other hand, our Inspection-L approach allows for every node to obtain access to the structural patterns of the entire graph, which can capture more global neighborhood information. The proposed method considers that the message-passing functions of [5] [16] are not powerful enough, as they lack injective functions. Therefore, we proposed a GIN encoder to make the message propagation function more robust.

3.1. Graph Neural Networks

GNNs is a deep learning approach for graph-based data and a recent and highly promising area of machine learning [23]. The key feature of GNNs is their ability to combine a topological graph structure with features. For each node in a graph, this means aggregating neighboring node features to leverage a new representation of the current node that considers the neighboring information. The output of this process is known as embeddings. Final node embeddings are low- or n-dimensional vector representations that capture topological and node properties. Embeddings can be learned in a supervised or unsupervised manner and used for downstream tasks such as node classification, clustering, and link prediction [23]. The k -th layer of a typical GCN is:

$$h_v^{(k)} = \sigma \left(W \cdot \text{MEAN} \{ h_u^{(k-1)}, \forall u \in N(v) \cup \{v\} \} \right). \quad (1)$$

where $h_v^{(k)}$ is the feature vector of node v at the k -th iteration/layer, $h_v^{(0)} = X_v$, and $N(v)$ is the set of neighbor nodes of v . $W^{(l)}$ is the weight matrix that will be learned for the downstream tasks. σ is an activation function, typically ReLU, for computing node representations.

3.2. Graph Isomorphism Network

Graph Isomorphism Network (GIN) is theoretically a maximally powerful GNN proposed by Xu et al. [24]. The main difference between GIN and other GNNs is the message aggregation function, which is shown below:

$$h_v^{(k)} = \text{MLP}^{(k)} \left(\left(1 + \epsilon^{(k)} \right) \cdot h_v^{(k-1)} + \sum_{u \in N(v)} h_u^{(k-1)} \right) \quad (2)$$

GCNs is less effective than the Weisfeiler–Lehman (1-WL) [22] test due to the single-layer aggregation function, which is same as the hash function of a 1-WL algorithm. According to [24], a single, non-linear layer is insufficient for graph learning. Thus, GCN message passing functions are not necessarily injective. Therefore, GIN [24] was proposed to make the passing function injective, as shown in Equation 2, where $\epsilon^{(k)}$ is a scalar parameter, and MLP stands for multilayer perceptron. $h_v^{(k)} \in \mathbb{R}^d$ is the embedding of node v_i at the k -th layer, $h_v^{(0)} = x_v$ is the original input node features, and $N(v_i)$ is the set of neighboring nodes of node v_i . We can stack k layers to obtain the final node representation $h_v^{(k)}$.

3.3. Deep Graph Infomax

Deep Graph Infomax (DGI) [4] is a self-supervised graph representation learning approach that relies on maximizing the mutual information between patch representations and the global graph summary. The patch representations summarize subgraphs, allowing for the preservation of similarities at the patch level. A trained encoder in DGI can be reused to generate node embeddings for downstream tasks, such as node clustering.

Most of the previous works on self-supervised representation learning approaches rely on the random walk strategy [25][26], which is extremely computationally expensive because the number of walks depends on the number of nodes on the graph, making it unscalable for large graphs. Moreover, the choice of hyperparameters (length of the walk, number of walks) can significantly impact the model performance. Overall, DGI does not require supervision or random walk techniques. Instead, it guides the model to learn node connections by simultaneously leveraging local and global information in a graph [4].

Figure 1 shows the overall operation of DGI. G is a true graph with the true nodes, the true edges that connect them, and real node features associated with each node. H is a corrupted graph where the nodes and edges have been changed using a corruption function. [4] suggests that the corruption function can randomly shuffle each node feature and maintain the same edges as the true graph G .

The DGI training procedure consists of four components:

- A corruption procedure C that changes the real input graph G into a corrupted graph $H = (C(G))$. This can be achieved by randomly shifting the node features among the nodes in a real graph G or by adding and removing an edge from the real graph G .
- An encoder E that computes the node embeddings of a corrupted graph and a real graph. This can be achieved using various graph representation methods, such as Graph Convolutional Networks (GCNs) [9], Graph Attention Networks (GATs) [27] or Graph Transformer Networks (GTNs) [28].
- The node embedding vectors for each node in the real graph are summarized into a single embed vector of the entire graph \vec{s} (global graph summary) by using a read-out function R to compute the whole graph embeddings.
- A discriminator D , which is a logistic non-linear sigmoid function, compares a real node embedding vector \vec{h}_i and a corrupted node embedding \tilde{h}_i against the whole real graph embedding \vec{s} , and provides a score between 0 and 1, as shown in Equation 3. This binary cross-entropy loss objective function [4] can be applied to discriminate between the embedding of the real node and the corrupted node to train the encoder E .

$$L = \frac{1}{N + M} \left(\sum_{i=1}^N \mathbb{E}_{(\mathbf{x}, \mathbf{A})} \left[\log D(\vec{h}_i, \vec{s}) \right] + \sum_{j=1}^M \mathbb{E}_{(\tilde{\mathbf{x}}, \tilde{\mathbf{A}})} \left[\log (1 - D(\vec{h}_j, \vec{s})) \right] \right) \quad (3)$$

4. PROPOSED METHOD

To construct Bitcoin transaction graphs from the dataset, we used 49 different Bitcoin transaction graphs (TGs) [5] using time steps so that the nodes can be represented as node transactions and the edges can be represented as flows of Bitcoin transactions. This is a very natural way to represent Bitcoin transactions.

The pseudocode and overall procedure of our proposed algorithm are shown in Algorithm 1 and Figure. 2. The proposed framework consists of two-stage: DGI training for node embedding extraction to perform feature augmentation, supervised machine learning classification.

4.1. DGI Training

To train the proposed model, the input includes the transaction graphs G with node features (i.e., all 166 features, which is a combination of local and macro features, which we denote AF, or only the 94 local features), and the specified number of training epochs K to extract true node embeddings and corrupted node embeddings. Before this, we need to define the corruption function C to generate the corrupted transaction graphs $C(G)$ for our GIN encoder to extract the corrupted node embeddings. In this paper, we randomly shuffled all the node features among the nodes in real transaction graphs G to generate the corrupted transaction graphs for each real graphs by shuffling the feature matrix in rows \mathbf{X} by using Bernoulli distribution. Overall, instead of adding or removing edges from the adjacency matrix such that $\mathbf{A}_G \neq \mathbf{A}_H$, we use corruption function C , which shuffle the node features such that $\mathbf{X}_G \neq \mathbf{X}_H$, and retain the adjacency matrix, i.e., $(\mathbf{A}_G = \mathbf{A}_H)$. Note that the corruption function only changes the node features, and not the structure; therefore, $N_G = N_H$. In case of the DGI implementation, we now have $N = M$.

For each batch of graph data G in the training epoch, in Algorithm 1 from Line 3 to 4, we use our proposed GIN encoder to extract true node and corrupted node embeddings. Our proposed GIN encoder is shown in Figure 2 with two layers of MLP, which consists of 128 hidden units, ReLU activation function and Batch normalization (as shown in Algorithm 2) [29].

The design of the MLPs is motivated by the fundamental goal of a GNN-based model. Ideally, various types of different graph patterns should be distinguishable via the graph encoder, which means that different graph structures should be mapped to different locations in the embedding space. This requires the

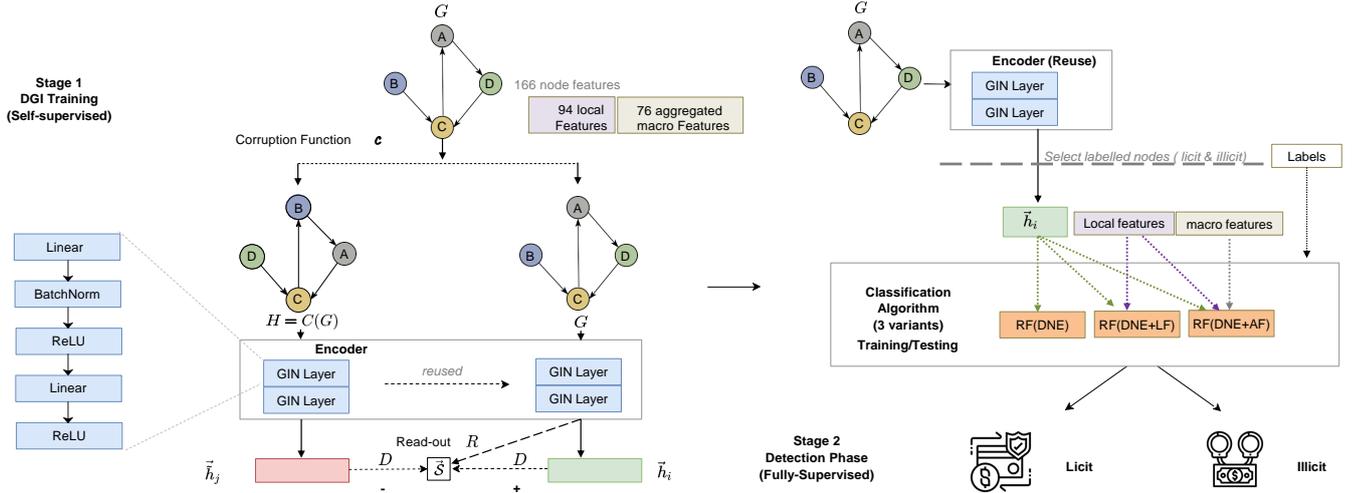


Figure 2: Proposed Method

Algorithm 1: Pseudocode for Our Proposed Algorithm

input : Set of training graphs $G^+ = \{G(V, A, X)\}$;
Number of training epochs K ;
Corruption function C ;
All 166 Features (AF);
First 94 Local Features (LF);

output: Optimized GIN encoder g , Optimized RF h_R

- 1 Initialize the parameters θ and ω for the encoder g and the discriminator D ;
- 2 **foreach** batch $G \in G^+$ **do**
- 3 **for** epoch $\leftarrow 1$ **to** K **do**
- 4 $h_i = g(G, \theta)$
- 5 $\tilde{h}_i = g(C(G), \theta)$
- 6 $\bar{s} = \sigma\left(\frac{1}{n} \sum_{i=1}^n h_i^{(L)}\right)$
- 7 $D(h_i, \bar{s}) = \sigma\left(h_i^T w \bar{s}\right)$
- 8 $D(\tilde{h}_i, \bar{s}) = \sigma\left(\tilde{h}_i^T w \bar{s}\right)$
- 9 $L_{DGI} = \frac{1}{N+M} \left(\sum_{i=1}^N \mathbb{E}_{(X,A)} \left[\log D(\vec{h}_i, \vec{s}) \right] + \sum_{j=1}^M \mathbb{E}_{(\bar{X}, \bar{A})} \left[\log (1 - D(\vec{h}_j, \vec{s})) \right] \right)$
- 10 $\theta, \omega \leftarrow \text{Adam}(L_{DGI})$
- 11 Select labeled node embedding h_i from $h_i = g(G, \theta)$ and corresponding labels y for $G \in$ training set;
- 12 $h_R \leftarrow \text{RF}(h_i || \{AF \text{ or } LF\}, y)$
- 13 **return** h_R, g

ability to solve the graph isomorphism problem, where non-isomorphic graphs should be mapped to different representations.

We applied a full neighbor sampling technique and used two-hop neighbor samples for the GIN encoder with Batch normalization, as DGI benefits from employing wider rather than deeper models [4].

For the read-out function R , we applied the mean operation on all node embeddings in the real graph G and then applied a sigmoid activation function to compute the whole graph embeddings \bar{s} :

$$\bar{s} = \sigma\left(\frac{1}{n} \sum_{i=1}^n h_i^{(L)}\right) \quad (4)$$

In Algorithm 1, from line 7 to 8, as shown in Equation 5 and Equation 6, for the discriminator D , we used a logistic sigmoid

Algorithm 2: Batch Normalizing Transform [29]

input : Values of x over a mini-batch: $B = \{x_1 \dots x_m\}$
Parameters can be learned: γ, β

output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

- 1 $\mu_B \leftarrow \frac{1}{m} \sum_{i=1}^m x_i$
- 2 $\sigma_B^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2$
- 3 $\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$
- 4 $y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i)$

non-linear function to discriminate node embedding vector \vec{h}_i against the real whole graph embedding \bar{s} to calculate the score of (\vec{h}_i, \bar{s}) being positive or negative:

$$D(h_i, \bar{s}) = \sigma\left(h_i^T w \bar{s}\right) \quad (5)$$

$$D(\vec{h}_i, \vec{s}) = \sigma(\vec{h}_i^T w \vec{s}) \quad (6)$$

We then used a binary cross-entropy loss objective function (based on Equation 3, modified so that $N = M$) to perform gradient descent, as shown in Algorithm 1, line 10. To perform gradient descent, we maximized the score if the node embedding is a true node embedding \vec{h}_i and minimized the score if it is a corrupted node embedding \vec{h}_i compared to the global graph summary generated by the read-out function R (Equation 4). As a result, we maximized the mutual information between patch representations and the whole real graph summary based on the binary cross-entropy loss function (BCE), as shown in Equation 3 to perform gradient descent. After the training process, the trained encoder can be used to generate new graph embeddings for downstream purposes; in this case, the detection of illegal transactions.

In our experiments, we used all 34 different Bitcoin transaction graphs to train the DGI with the GIN encoder in a self-supervised manner. For each training graph, we trained 300 epochs using an *Adam* optimizer with a learning rate of 0.0001, as shown in Algorithm 1, line 10.

4.2. Supervised Machine Learning Classification

After the DGI training, we reused the encoder to generate node embeddings, as shown in Algorithm 1, line 11 – 12 to train and test the RF classifier with 100 estimators. In our experiments, we performed 70:30 splitting, 34 different Bitcoin transaction graphs for training and the remaining 15 bitcoin transaction graphs for testing. All 34 training graphs were fed to DGI to train the GIN encoder in a self-supervised manner. Once the training phase was completed, we used a trained GIN encoder to extract all the node embeddings (all 34 graph node embeddings) in the training graphs. As the datasets consist of two labels, binary classification and unknown labels, we dropped unknown label data in the RF training and testing phases and only used label data for performance. We used all training graph node embeddings to train the RF in a supervised manner. For testing, we extracted the last 15 test graph node embeddings using the trained GIN and fed the node embeddings to the trained RF for illegal transaction detection.

We experimented with the following three combinations of features and embeddings:

1. **DNE : Node Embeddings only:** After the DGI training, we reused the encoder to generate node embeddings for training and testing the RF classifier, as mentioned above.
2. **LF + DNE : Node Embeddings with LF features:** Similar to scenario 1, we also combined local features (i.e, first 94 raw features) with the node embeddings generated by the trained encode for training and testing the RF classifier, as mentioned above.
3. **AF + DNE : Node Embeddings with AF Features:** Similar to scenario 1, we also combined all raw features (AF features) with the node embeddings generated by the trained encoder for training and testing the RF classifier, as mentioned above.

Table 1: Implementation environment specification

Unit	Description
Processor	2.3 GHz 2-core Inter Xeon(R) Processor
RAM	12GB
GPU	Tesla P100 GPU 16GB
Operating System	Linux
Packages	Sckit-learn, Numpy, Pandas, PyTorch Geometric, and Matplotlib

4.3. Implementation Environments

Experiments were carried out using a 2.3GHz 2-core Intel(R) Xeon(R) processor with 12 GB memory and Tesla P100 GPU on a Linux operating system. The proposed approach was developed using the Python programming language with several statistical and visualization packages, such as Sckit-learn, Numpy, Pandas, PyTorch Geometric, and Matplotlib. Table 1 summarizes the system configuration.

5. Experiments and Results

5.1. Dataset

In this paper, we adopted the Elliptic dataset [5], which is the world’s largest labeled dataset of bitcoin transactions. The Elliptic dataset [5] consists of 203,769 node as transactions and 234,355 directed transaction payment flows (i.e., transaction inputs, transaction outputs). The datasets also consists of 49 different timestep graphs, which are uniformly spaced with a two-week interval, as illustrated in 3. Each connected transaction component consists of a time step that appears on the blockchain in less than three hours. Our G represents one such transaction graph for the 49.

In the Elliptic dataset [5], 21% of the node entities are labeled as licit, and only 2% are labeled as illicit. The remaining node entities are unlabeled but have node features. These node entities consist of 166 features (AF features), among which the first 94 features contain local information (LF features) of the transactions, including the time step, transaction fees, and the number of inputs or outputs. The remaining 72 features are aggregated features. These features can be obtained by aggregating transaction information from one-hop backward/forward graph nodes, such as the standard deviation, minimum, maximum, and correlation coefficients of the neighbor transactions for the same information data. More importantly, all features were obtained using only publicly available information.

5.2. Performance Metric

To evaluate the performance of the proposed methods, the standard metrics listed in Table 2 were used, where TP , TN , FP and FN represent the number of True Positives, True Negatives, False Positives and False Negatives, respectively.

In Table 2, true positive (TP) denotes the total number of true positives, true negative (TN) indicates the total number of false positives, false positive (FP) denotes the total number of false negatives and false negative (TN) shows the total number of true negatives. The proposed method was evaluated using

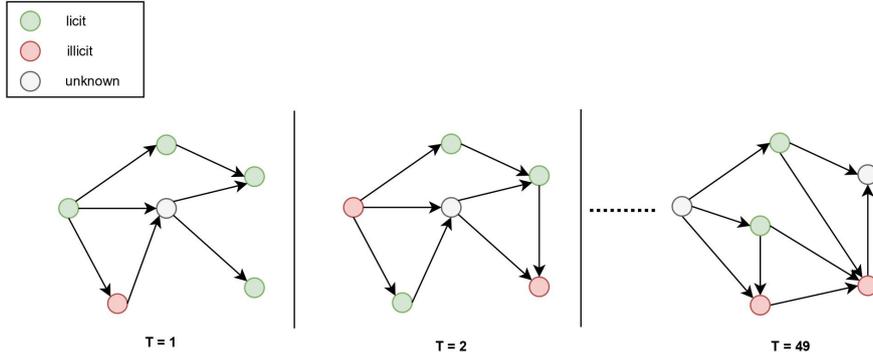


Figure 3: Overview of Elliptic Dataset [30]

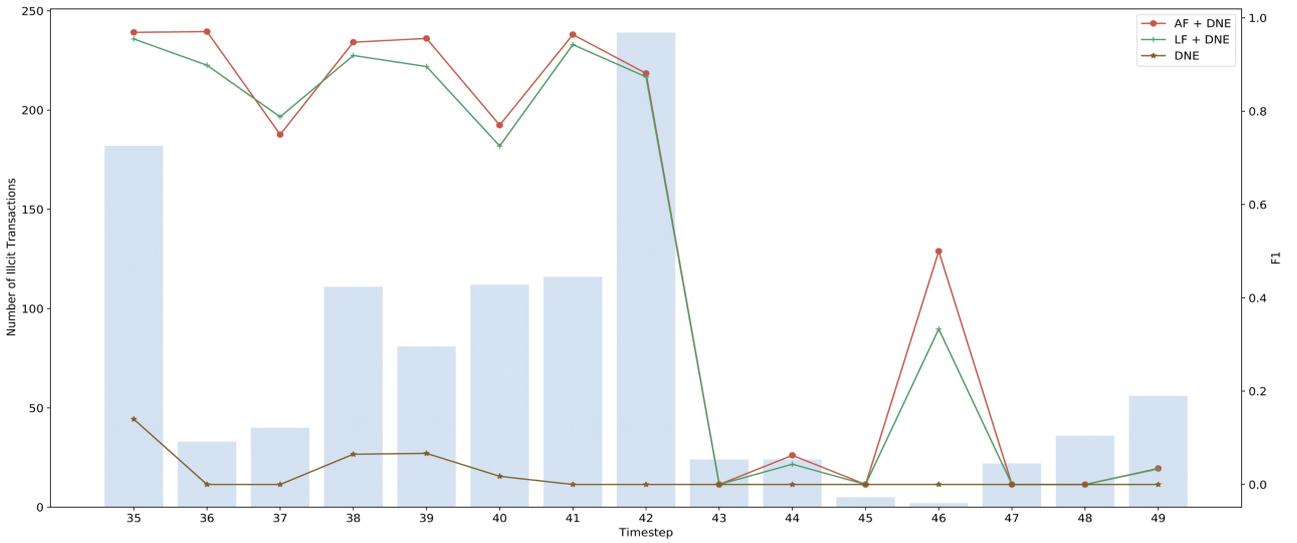


Figure 4: Illicit F1 over test timestep

Table 2: Evaluation metrics used in this study

Metric	Definition
Detection Rate (Recall)	$\frac{TP}{TP+FN}$
Precision	$\frac{TP}{TP+FP}$
F1-Score	$2 \times \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$
AUC-Score	$\int_0^1 \frac{TP}{TP+FN} d \frac{FP}{TN+FP}$

Precision, Recall, F1-score and Area under the receiver operating characteristics (ROC) curve. All the above metrics can be obtained using the confusion matrix (CM).

Accuracy indicates that the model is well learned in case of a balanced test dataset; however, for imbalanced scenarios, as in this case, only considering accuracy measures may lead to misleading conclusion,s since it is strongly biased in favor of the licit majority class. Thus, for this case, recall and F1-score metrics provide a more reasonable explanation of the model’s performance.

Recall (also known as Detection Rate) is the total number of

true positives divided by the total number of true positives and false negatives. If the recall rate is very low, this means that the classifier cannot detect illicit transactions.

Precision measures the quality of the correct predictions. This is the number of true positives divided by the number of true positives and false positives. If the false positive is very high, it will cause low precision. Our goal is to maximize the precision as much as possible.

F1-score is the trade-off between precision and recall. Mathematically, it is the harmonic mean of precision and recall.

The area under the curve (AUC) computes the trade-off between sensitivity and specificity, plotted based on the trade-off between the true positive rate on the y-axis and the false positive rate on the x-axis. Our goal is to maximize the AUC score as much as possible, making is closer to 1.0.

5.3. Experimental Results

Table 3 shows the corresponding results of our Inspection-L compared to the state-of-the-art in terms of the key metrics. As can be observed from the table, regarding to illicit F1-Score, Inspection-L (LF+DNE and AF+DNE) outperforms the best

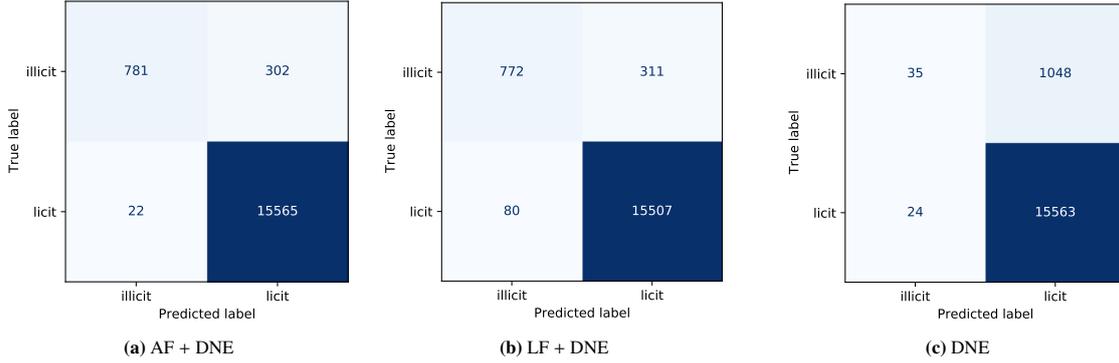


Figure 5: Confusion Matrix

Table 3: Results of binary classification by Inspection-L compared to the state-of-the-art. AF refers to all raw features, LF refers to the local raw features, i.e., the first 94 raw features, GNE refers to the node embeddings generated by GCN in [5] using labels and DNE refers to the node embeddings computed by DGI without using labels.

Method	Illicit			AUC
	Precision	Recall	F1	
Logistic Regr ^{AF} [5]	0.404	0.593	0.481	–
Logistic Regr ^{AF + GNE} [5]	0.537	0.528	0.533	–
Logistic Regr ^{LF} [5]	0.348	0.668	0.457	–
Logistic Regr ^{LF + GNE} [5]	0.518	0.571	0.543	–
RandomForest ^{AF} [5]	0.956	0.670	0.788	–
RandomForest ^{AF + GNE} [5]	0.971	0.675	0.796	–
RandomForest ^{AF} [14]	0.897	0.721	0.800	–
RandomForest ^{AF + GNE} [14]	0.958	0.715	0.819	–
XGB ^{AF} [14]	0.921	0.732	0.815	–
XGB ^{AF + GNE} [14]	0.986	0.692	0.813	–
RandomForest ^{LF} [5]	0.803	0.611	0.694	–
RandomForest ^{LF + GNE} [5]	0.878	0.668	0.759	–
MLP ^{AF} [5]	0.694	0.617	0.653	–
MLP ^{AF + GNE} [5]	0.780	0.617	0.689	–
MLP ^{LF} [5]	0.637	0.662	0.649	–
MLP ^{LF + GNE} [5]	0.681	0.578	0.625	–
GCN [5]	0.812	0.512	0.628	–
GCN [16]	0.899	0.678	0.773	–
Skip-GCN [5]	0.812	0.623	0.705	–
EvolveGCN [5]	0.850	0.624	0.720	–
Inspection-L ^{DNE} (RF)	0.593	0.032	0.061	0.735
Inspection-L ^{LF + DNE} (RF)	0.906	0.712	0.797	0.895
Inspection-L^{AF + DNE} (RF)	0.972	0.721	0.828	0.916

reported classifiers. In the best-performing variant, AF+DNE, we concatenated the node embeddings generated from DGI with all original raw features (AF). The experiment achieved an F1 score and Recall of 0.828 and 0.721, respectively. Using all features (AF) with node embeddings (DNE) as input for classification, the ML model’s performance significantly increased, with an AUC of 0.916, compared to 0.735 when only the node embeddings were used for classification. The experiments demonstrate that graph information (node embeddings) is useful to enhance the transaction representations (embeddings).

In the second experiment LF+DNE, we concatenated the node embeddings generated from DGI with the local features (LF), which can achieve an F1-score and Recall of 0.712 and 0.797, respectively. Both the results were superior to the state-of-the-art algorithms.

These results demonstrate the ability of our self-supervised GIN-based approach to generate an enhanced feature set to improve anti-money-laundering detection performance. Furthermore, the results show that the accuracy of the model improves with the enhanced feature set, which contains summary information. Note that the summary information in the AF feature set consists of 1-hop forward and 1-hop backward neighborhood summaries for each node. Unfortunately, the Elliptic dataset does not provide detailed information regarding the feature descriptions, possibly due to confidentiality reasons, which limits our ability to provide a deeper discussion.

Figure 4 shows the F1 measure of the three different model variants across various testing timesteps. Interestingly, none of the three variants can detect new illicit transactions with high precision after dark market shutdown, which occurs at time step 43 [5]. Thus, we note that developing robust methods to detect illicit transactions without their being affected by emerging events is a major challenge that future works need to address.

Figure 5 shows the confusion matrix of the three different scenarios. Although the classifier trained with embedding features cannot accurately detect illicit transactions, it rarely classifies licit transactions as illicit. Therefore, the false alarm rate is very low, as shown in Figure 5c. The RF classifier trained using both raw features and embedding features, shown in Figure 5a,5b, has the advantage of achieving a high detection rate and a low false alarm rate. As a result, the experimental results demonstrate that DNE node embeddings can be used for feature augmentation to improve overall detection performance.

5.4. Broader applications of AML

The blockchain operates as a decentralized bank for bitcoin cryptocurrency [31]. All bitcoin transactions are permanently recorded on the blockchain, which is a visible and verifiable public ledger [32]. Bitcoin addresses are not registered to individuals, in contrast to bank accounts [2]. Thus, due to this pseudo-anonymity [11], bitcoin and other crypto-currencies are increasingly used for ransomware [2], ponzi schemes [11] and illicit material trade on the dark web [23]

While bitcoin transactions are difficult to track, they are not completely anonymous [2]. Users can be traced by their IP addresses and transaction flows [32]. An analysis of the bitcoin graph can reveal suspicious behavior patterns characteristic of

money laundering [2]. To break the tell-tale transnational link between bitcoin transactions and illegal activity, bitcoin mixing services provide a new, untainted bitcoin address from their reserves and the pay-outs are spread out over time [2]. Bitcoin Fog is a service that hides transaction origins by bundling multiple inputs into a smaller number of larger outputs [11]. However, the additional obscuring activities themselves could add characteristic signatures into transaction flows. Thus, it is still possible to detect patterns in the underlying transaction flow to facilitate AML detection [11, 5]. Unfortunately, next-generation cryptocurrencies such as Monero, Dash, and Z-Cash, with built-in anonymity features, make tracking and detection challenging [2]. As a result, there is a constant need for improved AML detection methodologies.

6. Conclusions and Future Work

This paper presents a novel approach for the detection of illicit Bitcoin transactions based on self-supervised GNNs. We first used the DGI to generate the node embedding with raw features to train the Random Forest for detection. Our experimental evaluation indicates that our approach performs exceptionally well and outperforms the state-of-the-art ML-based/Graph-based classifier overall. The evaluation results of our initial classifier demonstrate the potential of using a self-supervised GNN-based approach for illegal transaction detection in cryptocurrencies. We hope to inspire others to work on the important challenge of using graph machine learning to perform financial forensics through this research, which is lacking in the current research. In the future, we plan to integrate this with unsupervised anomaly detection algorithms to detect illegal transactions in an unsupervised manner.

References

- [1] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, Technical Report, Manubot, 2019.
- [2] N. Kshetri, J. Voas, Do crypto-currencies fuel ransomware?, in: IT professional, volume 19, IEEE, 2017, pp. 11–15.
- [3] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P. S. Yu, A comprehensive survey on graph neural networks, in: IEEE Transactions on Neural Networks and Learning Systems, volume 32, 2021, pp. 4–24.
- [4] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, R. D. Hjelm, Deep graph infomax, in: International Conference on Learning Representations, 2019. URL: <https://openreview.net/forum?id=rk1z9iAcKQ>.
- [5] M. Weber, G. Domeniconi, J. Chen, D. K. I. Weidele, C. Bellei, T. Robinson, C. E. Leiserson, Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics, in: ACM SIGKDD International Workshop on Knowledge discovery and data mining, 2019.
- [6] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, J. Tang, Self-supervised learning: Generative or contrastive, in: IEEE Transactions on Knowledge and Data Engineering, 2021, pp. 1–1. doi:10.1109/TKDE.2021.3090866.
- [7] Y. Liu, M. Jin, S. Pan, C. Zhou, Y. Zheng, F. Xia, P. Yu, Graph self-supervised learning: A survey, in: IEEE Transactions on Knowledge and Data Engineering, 2022, pp. 1–1. doi:10.1109/TKDE.2022.3172903.
- [8] C. M. Bishop, N. M. Nasrabadi, Pattern recognition and machine learning, volume 4, Springer, 2006.
- [9] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, 2017.
- [10] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, C. Leiserson, Evolvegcn: Evolving graph convolutional networks for dynamic graphs, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pp. 5363–5370.
- [11] Y. Hu, S. Seneviratne, K. Thilakarathna, K. Fukuda, A. Seneviratne, Characterizing and detecting money laundering activities on the bitcoin network, arXiv preprint arXiv:1912.12060 (2019).
- [12] J. A. Bondy, U. S. R. Murty, et al., Graph theory with applications, volume 290, Macmillan London, 1976.
- [13] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.
- [14] D. Vassallo, V. Vella, J. Ellul, Application of gradient boosting algorithms for anti-money laundering in cryptocurrencies, in: SN Computer Science, volume 2, Springer, 2021, pp. 1–15.
- [15] C. Lee, S. Maharjan, K. Ko, J. W.-K. Hong, Toward detecting illegal transactions on bitcoin using machine-learning methods, in: International Conference on Blockchain and Trustworthy Systems, Springer, 2019, pp. 520–533.
- [16] I. Alarab, S. Prakoonwit, M. I. Nacer, Competence of graph convolutional networks for anti-money laundering in bitcoin blockchain, in: Proceedings of the 2020 5th International Conference on Machine Learning Technologies, 2020, pp. 23–27.
- [17] L. Nan, D. Tao, Bitcoin mixing detection using deep autoencoder, in: 2018 IEEE Third international conference on data science in cyberspace (DSC), IEEE, 2018, pp. 280–287.
- [18] J. Lorenz, M. I. Silva, D. Aparício, J. T. Ascensão, P. Bizarro, Machine learning methods to detect money laundering in the bitcoin blockchain in the presence of label scarcity, in: Proceedings of the First ACM International Conference on AI in Finance, 2020, pp. 1–8.
- [19] T. Pham, S. Lee, Anomaly detection in bitcoin network using unsupervised learning methods, in: arXiv preprint arXiv:1611.03941, 2016.
- [20] P. Monamo, V. Marivate, B. Twala, Unsupervised learning for robust bitcoin fraud detection, in: 2016 Information Security for South Africa (ISSA), IEEE, 2016, pp. 129–134.
- [21] S. Li, F. Xu, R. Wang, S. Zhong, Self-supervised incremental deep graph learning for ethereum phishing scam detection, in: arXiv preprint arXiv:2106.10176, 2021.
- [22] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, K. M. Borgwardt, Weisfeiler-lehman graph kernels., in: booktitle of Machine Learning Research, volume 12, 2011.
- [23] G. K. Kulatilleke, M. Portmann, R. Ko, S. S. Chandra, Fdgatii: Fast dynamic graph attention with initial residual and identity mapping, in: arXiv preprint arXiv:2110.11464, 2021.
- [24] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks?, in: International Conference on Learning Representations, 2019. URL: <https://openreview.net/forum?id=ryGs6iA5Km>.
- [25] W. L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems, 2017. arXiv:1706.02216.
- [26] C. Zhang, D. Song, C. Huang, A. Swami, N. V. Chawla, Heterogeneous graph neural network, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 793–803.
- [27] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations (ICLR), 2018.
- [28] S. Yun, M. Jeong, R. Kim, J. Kang, H. J. Kim, Graph transformer networks, in: Advances in neural information processing systems, volume 32, 2019.
- [29] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International conference on machine learning, PMLR, 2015, pp. 448–456.
- [30] D. T. Robinson, How to Combat Financial Crime in Cryptocurrencies, 2019. URL: <https://www.elliptic.co/blog/elliptic-dataset-cryptocurrency-financial-crime>.
- [31] S. Nakamoto, Bitcoin: a peer-to-peer electronic cash system [eb/ol], Consulted 1 (2008) 28.
- [32] R. Van Wegberg, J.-J. Oerlemans, O. van Deventer, Bitcoin money laundering: mixed results? an explorative study on money laundering of cybercrime proceeds using bitcoin, Journal of Financial Crime (2018).