

Article

HBTD: A Heterogeneous Bitcoin Transaction Behavior Dataset for Anti-Money Laundering

Jialin Song and Yijun Gu *

School of Information Network Security, People's Public Security University of China,
Beijing 100032, China; 2020211634@stu.ppsuc.edu.cn

* Correspondence: guyijun@ppsuc.edu.cn

Abstract: In this paper, we predict money laundering in Bitcoin transactions by leveraging a deep learning framework and incorporating more characteristics of Bitcoin transactions. We produced a dataset containing 46,045 Bitcoin transaction entities and 319,311 Bitcoin wallet addresses associated with them. We aggregated this information to form a heterogeneous graph dataset and propose three metapath representations around transaction entities, which enrich the characteristics of Bitcoin transactions. Then, we designed a metapath encoder and integrated it into a heterogeneous graph node embedding method. The experimental results indicate that our proposed framework significantly improves the accuracy of illicit Bitcoin transaction recognition compared with traditional methods. Therefore, our proposed framework is more conducive in detecting money laundering activities in Bitcoin transactions.

Keywords: graph embedding; Bitcoin anti-money laundering; machine learning; heterogeneous graph neural networks; metapath encoder



Citation: Song, J.; Gu, Y. HBTD: A Heterogeneous Bitcoin Transaction Behavior Dataset for Anti-Money Laundering. *Appl. Sci.* **2023**, *13*, 8766. <https://doi.org/10.3390/app13158766>

Academic Editors: José Salvador Sánchez Garreta and Gianluca Lax

Received: 12 June 2023

Revised: 12 July 2023

Accepted: 27 July 2023

Published: 29 July 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Bitcoin money laundering refers to the illegal use of Bitcoin to transfer funds covertly and obscure the source of those funds, taking advantage of Bitcoin's decentralization and anonymity [1].

The graph neural network (GNN) has emerged in recent years. Researchers prefer GNN methods rather than traditional machine learning methods to study Bitcoin trading networks. These are the advantages:

- (1) Graph modeling capability: Graph neural networks excel at handling graph structures, allowing them to effectively analyze the Bitcoin transaction network.
- (2) Node embedding learning: Graph neural networks learn compact representations for Bitcoin addresses, enabling tasks like clustering, classification, and prediction.
- (3) Integrating multi-level information: Graph neural networks seamlessly incorporate various types of information, such as structural and attribute information of each Bitcoin transaction.

As the work on Bitcoin transaction detection using GNN has gradually increased, the existing related studies can be mainly divided into three types: random walk, graph convolution network, and graph self-encoding. However, each of these methods comes with its own set of advantages and disadvantages:

- (1) Random walk: It can capture local neighbor relations of nodes but has sampling bias and information loss on sparse graphs.
- (2) Graph convolution network: It can capture the topology between nodes and fuse the characteristic information of nodes, but it cannot capture the long-distance dependencies in the graph [2], and when dealing with unbalanced datasets, the message transmission mechanism will ignore the illegal nodes [3].
- (3) Graph autoencoder: It can handle unsupervised and semi-supervised tasks, but overfitting may occur in cases of scarce samples.

As for the perspective of the dataset, many challenges and limitations have emerged, from label loss to anonymity (e.g., certain dataset attribute information is not clear). Another limitation we observe is the lack of a heterogeneous dataset that can be used as a research baseline because we believe that the wallet address and transaction entity are equally important in the Bitcoin transaction network structure. Therefore, based on the Elliptic [4] dataset, we reconstructed the Bitcoin transaction dataset from the perspective of heterogeneous graphs to start our work. At the same time, we provided a de-anonymized Bitcoin transaction hash and Bitcoin wallet address to facilitate researchers to jointly promote related studies together.

In summary, we propose a framework for Bitcoin anti-money laundering detection. From the collection of Bitcoin transaction data, the extraction of the metapath in the transaction network, to the model training on GNN, and finally, through the classifier, the accuracy of identifying Bitcoin money laundering behavior is improved. The idea of this framework can be extended to the practical application of Bitcoin regulations because its advantages are the flexibility and information richness of the model. It is convenient to improve the encoder, GNN, and classifier parts of this framework. At the same time, when the Bitcoin wallet address is associated with the Bitcoin transaction entities, it can detect the legitimacy of the Bitcoin transaction entities as well as grasp the legitimacy of the wallet address participating in the transaction, which will be beneficial to the relevant regulatory authorities.

In general, our contributions are as follows:

(1) We constructed the HBTBD (heterogeneous bitcoin transaction behavior dataset), which further enriched the elliptic datasets by collecting historical transaction data on the blockchain. We introduce wallet address nodes based on Bitcoin transaction entities, extracted multiple metapath information, and make it available on Kaggle (<https://www.kaggle.com/datasets/songjialin/hbtbd-for-aml>, accessed on 26 July 2023). The code used to build the HBTBD is posted on GitHub (<https://github.com/Tsunami-Song/HBTBD>, accessed on 26 July 2023).

(2) We designed a metapath encoder that aims to encode the node features of the metapath better through TransE [5].

(3) Through experiments, we proved that building a heterogeneous transaction network model based on the metapath aggregated graph neural network (MAGNN) [6] helps improve the performance of detecting Bitcoin money laundering for the same Bitcoin transaction behavior. Also, the model can be considered a baseline method for the heterogeneous network dataset. It provides more possibilities for future research perspectives.

2. Related Work

There are many studies on Bitcoin anti-money laundering, including work on using well-known supervisory learning models, such as support vector machine (SVM) [7], random forest [8], and XGBoost [9]. However, due to the simple structure of the model, it cannot capture the information in the Bitcoin transaction network structure, so it has a poor effect on money laundering detection.

In 2019, Weber et al. [10] first proposed the elliptic dataset and attempted to use graph embedding methods, such as graph convolutional networks (GCNs) [11], to detect money laundering transactions. Aldo Pareja et al. [12] used EvolveGCN to adjust the graph convolution model along the time dimension; they evolved GCN parameters through long short-term memory (LSTM) [13] and gated recurrent units (GRUs) [14] to capture the dynamics of graph sequences, proving the advantages of EvolveGCN in the elliptic dataset. LSTM is used for the EvolveGCN-O method, where the GCN parameters are inputs/outputs of a recurrent architecture, and GRU is used for the EvolveGCN-H, where the GCN parameters are hidden states of a recurrent architecture that takes node embeddings as input. Hu et al. [15] used various graph features on the Bitcoin transaction graph to distinguish money laundering transactions from regular transactions. They showed that classifiers based on different feature types, namely direct neighbor, curated feature, DeepWalk [16] embedding, and node2vec [17] embedding, are significantly superior to

manually created features. Lorenz et al. [18] proposed active learning techniques to study the minimum number of tags necessary to achieve high detection and test their illegal activity datasets. In addition, Alarab et al. [19] presented an integrated learning method using a given combination of supervised learning models and applied it to the elliptic dataset, improving the baseline results. Jensen et al. [20] used gated recurrent units and a self-attention model, which can reduce the number of false positives when qualifying alarms. Yang et al. [21] effectively discerned conspicuous anomalous money laundering patterns across diverse transaction datasets by training a combined long short-term memory and graph convolutional neural network model. Zhao et al. [22] combined mutual information and self-supervised learning to create a model based on mutual information that is used to improve the massive amount of untagged data in the dataset. In 2022, Yuexin Xiang et al. [23] proposed a classification framework based on a heterogeneous graph of Bitcoin transaction behavior, integrated the information of transaction nodes, extracted relatively rich features, and used traditional machine learning methods to classify Bitcoin wallet addresses. Inspired by this, this paper, based on the elliptic anti-money laundering dataset, constructs a heterogeneous graph dataset with transaction entities as the target nodes and uses the latest heterogeneous graph embedding model to verify the feasibility of the framework.

3. Preliminary

Unspent transaction output (UTXO) is the core of Bitcoin transactions. The graph structure can intuitively represent the UTXO model of Bitcoin transactions, including the relationship between the input and output of each Bitcoin transaction, and their links to previous transactions. Researchers have proposed a variety of graph model designs for different datasets to sufficiently express the information on Bitcoin transactions. Most analyze the transaction entity or wallet address as the target node because they contain more information.

To better aggregate the characteristics of transaction entities and wallet addresses and fully reflect the characteristics of keeping Bitcoin transactions flowing, we built a heterogeneous Bitcoin transaction network with node attributes, as shown in Figure 1. We take the Bitcoin transaction entity Tx_i as the main research object, adding the Bitcoin wallet address node Add_i and its characteristics to the graph. Finally, the graph embedding on the whole heterogeneous attribute graph is realized to discover the characteristics of those Bitcoin transaction entities that commit money laundering.

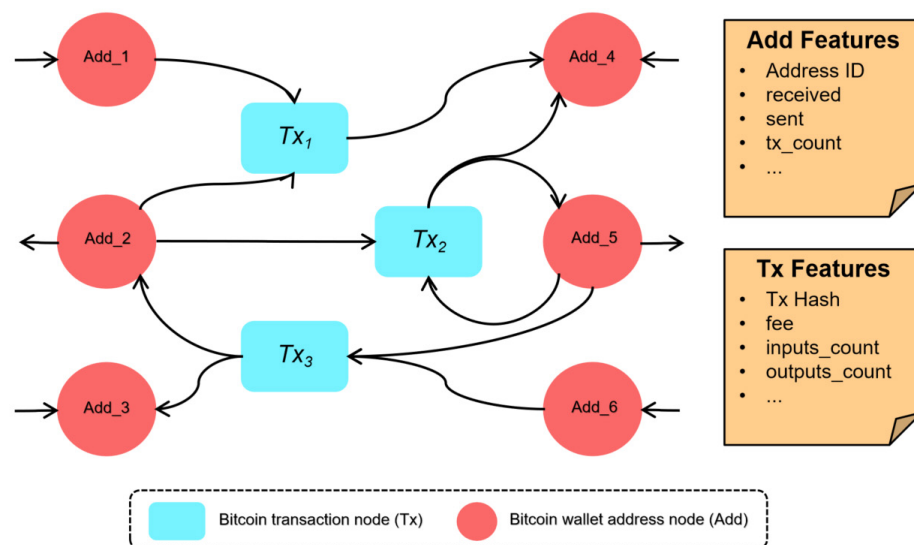


Figure 1. A Bitcoin transaction heterogeneous network.

3.1. Dataset Background

The elliptic dataset describes a network that represents Bitcoin transactions. The dataset classifies the labels of the transaction entities into three categories: “licit”, “illicit”, and “unknown”. Licit types include exchanges, wallet providers, miners, and financial service providers. Illicit categories include fraud, malware, terrorist organizations, and Ponzi schemes. Although the dataset provides 166 characteristics of all transaction entities, these characteristics are anonymous. Moreover, hashes of each trading entity are also hidden, limiting the extension of other characteristics related to transaction nodes.

Alexander [24] successfully de-anonymized node IDs in the elliptic dataset through feature analysis and published the correspondence between node IDs in the original dataset and real transaction hash in Kaggle (<https://www.kaggle.com/datasets/alexbenzik/deanonymized-995-pct-of-elliptic-transactions>, accessed on 26 July 2023). It lays a foundation for constructing this paper’s heterogeneous graph dataset.

3.2. Data Collection

We constructed the HBTBD dataset; the label was derived from the information of 46,045 transaction entities at 49 timesteps in the elliptic dataset. Among all transactions, 9.87% (4545 transactions) were marked as illicit and the remaining 90.13% (41,500 transactions) were marked as licit, as shown in Figure 2. We discard the data labeled “unknown” from the original dataset because we hope to detect illicit transactions in a supervised way. Bitcoin transactions are entirely transparent, and each transaction is recorded on the network’s public ledger. Thus, we use the query function of Bitcoin blockchain transaction data provided by BTC (<https://btc.com/>, accessed on 26 July 2023) for data collection. To expand the node types on the graph, we collected 319,311 Bitcoin wallet addresses and fundamental characteristics that participated in the input and output of these transactions by using the Bitcoin transaction API (<https://chain.api.btc.com/v3/tx/>, accessed on 26 July 2023).

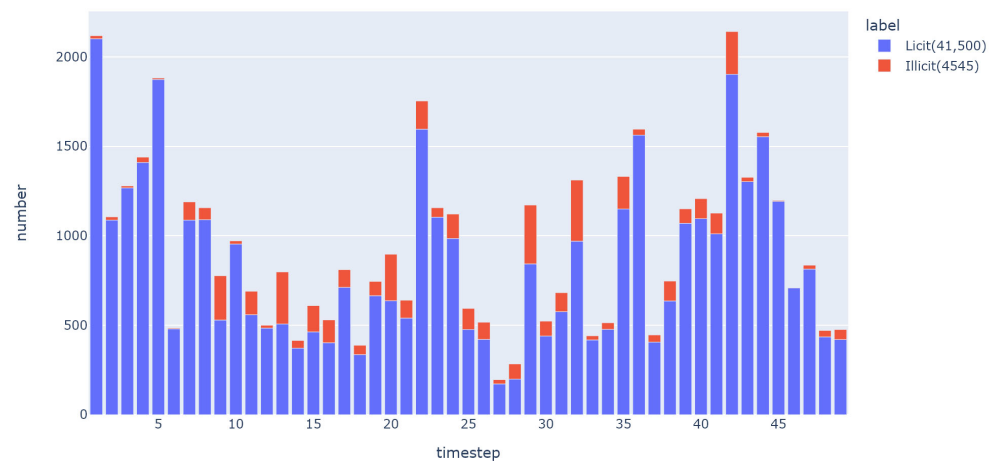


Figure 2. Distribution of Bitcoin transaction labels.

For the collection of wallet addresses, we use Python’s WebDriver module to implement the simulation browser. Accessing the txHash of the corresponding transaction through the API interfaces returns a response in JSON format, including the input and output wallet addresses.

To collect wallet address features, we use Bitcoin address API (<https://chain.api.btc.com/v3/address/>, accessed on 26 July 2023) and obtain the corresponding information about the wallet address through the interface. From the data, we extracted eight basic features: wallet income, wallet expenditure, wallet balance, number of transactions participated in by wallet, number of undetermined transactions, undetermined income,

undetermined spending, and number of unspent transactions, for the construction of the dataset.

3.3. Main Concept

This section formally defines some essential terminologies related to heterogeneous graphs. Table 1 summarizes frequently used notations in this paper for quick reference.

Table 1. Notations and Explanations.

Notations	Explanations
N	Number of Bitcoin transaction entity nodes
M	Number of wallet address nodes
D	Number of node features
Tx_i	Bitcoin transaction entity node i
Add_i	Bitcoin wallet address node i
$\mathcal{V}_{Tx}$	The set of Bitcoin transaction entity nodes in a graph
$\mathcal{V}_{Add}$	The set of wallet address nodes in a graph
$\mathcal{E}$	The set of edges in a graph
$\mathcal{G}$	A graph $\mathcal{G} = (\mathcal{V}_{Tx}, \mathcal{V}_{Add}, \mathcal{E}, \mathcal{C})$
P	Metapath
$P(Tx_v, Add, Tx_u)$	Metapath between node Tx_v and Tx_u connected by Add
h_i	Attributes or embeddings of node
M_r	Relation-specific matrix of relation
σ	Sigmoid function
$\odot$	Hadamard product
$\oplus$	Concatenation operator

Definition 1. *Bitcoin transaction heterogeneous information network.* A Bitcoin transaction heterogeneous information network is typically represented by a graph, as $\mathcal{G} = (\mathcal{V}_{Tx}, \mathcal{V}_{Add}, \mathcal{E}, \mathcal{C})$, where $\mathcal{V}_{Tx} = \{v_1, v_2, \dots, v_N\}$ is the set of all Bitcoin transaction entity nodes and $\mathcal{V}_{Add} = \{v_{N+1}, v_{N+2}, \dots, v_{N+M}\}$ is the set of all Bitcoin wallet address nodes in the graph. It is a bipartite graph, meaning all nodes can be divided into two sets, and there are no edges between the points of the same set. $\mathcal{E} \subset \{(v_i, v_j) \mid v_i \in \mathcal{V}_{Tx}, v_j \in \mathcal{V}_{Add}\}$ is the set of edges between transaction nodes and wallet address nodes. Since there is an input–output relationship between the Bitcoin transaction entity and the wallet address, the Bitcoin information transaction graph is directed. $\mathcal{C}$ denotes the node attribute feature, which is an $N \times D$ matrix with C_i as rows, where $C_i \in \mathbb{R}^D$ is the attribute vector associated with the node $Tx_i \in \mathcal{V}_{Tx}$. For different types of nodes, D represents the number of attribute features it has.

Definition 2. *Bitcoin transaction metapath.* Metapath is a concept that Yizhou Sun et al. [25] put forward in 2011. A metapath is an ordered sequence of node types and edge types defined in a graph that describes the composite relationship between the node types involved.

This paper defines a Bitcoin transaction metapath P as a path in the form of $Tx_v \xrightarrow{r_1} Add \xrightarrow{r_2} Tx_u$ (abbreviated as $Tx_v - Add - Tx_u$), which describes a composite relation $R = r_1 \circ r_2$ between two Bitcoin transaction nodes. For this, the length of the Bitcoin transaction path is constant at 3.

3.4. Bitcoin Wallet Address Node-Type Division

In the heterogeneous information network of Bitcoin transactions, Bitcoin transaction entities and Bitcoin wallet addresses are represented by nodes in the graph structure. In the anti-money laundering detection of Bitcoin, we aim to detect the legitimacy of the transaction node, so we mark the transaction node as Tx . In addition, we further divide the wallet address node into three categories by analyzing the characteristics of the wallet address participating in the transaction, as shown in Figure 3.

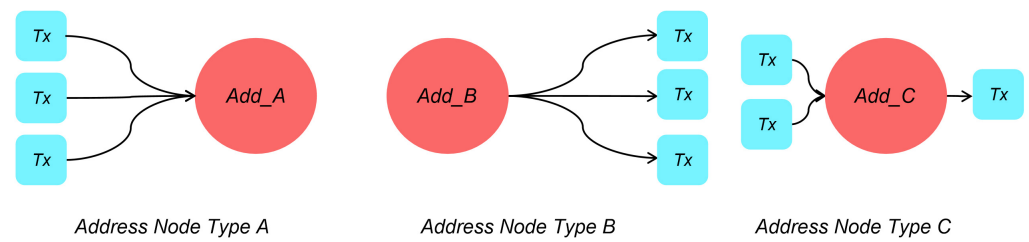


Figure 3. Bitcoin wallet address types.

It is worth noting that the premise of dividing the Bitcoin wallet address types involves the situation under an independent timestep; so for the wallet, there are three states: input only, output only, and input and output coexisting.

Algorithm 1 describes the node-type division method, the pseudocode is shown below.

Given the Bitcoin transaction heterogeneous information network $\mathcal{G}$, the wallet address node is divided into three categories: For the wallet address node with only input and no output, which is when $Add^{in} > 0$ and $Add^{out} = 0$, marked as Add_A (line 2, 3); for the wallet address node with only output and no input, which is when $Add^{out} > 0$ and $Add^{in} = 0$, marked as Add_B (lines 5, 6); for the wallet address node with both input and output records, which is when $Add^{in} > 0$ and $Add^{out} > 0$, marked as Add_C (lines 8, 9).

Algorithm 1 Bitcoin Wallet Address Node-Type Division

Input: The Bitcoin address node data $Addnode_List$

Output: Address node of various types: Add_A , Add_B , Add_C

```

1: for each  $addnode$  in  $Addnode\_List$  do
2:   if  $addnode.outdegree == 0$  and  $addnode.indegree != 0$  then
3:      $Add\_A.append(addnode)$ 
4:   end if
5:   if  $addnode.outdegree != 0$  and  $addnode.indegree == 0$  then
6:      $Add\_B.append(addnode)$ 
7:   end if
8:   if  $addnode.outdegree != 0$  and  $addnode.indegree != 0$  then
9:      $Add\_C.append(addnode)$ 
10:  end if
11: end for

```

To summarize, in this Bitcoin transaction framework, we designed four node types: Tx , Add_A , Add_B , and Add_C to participate in constructing a heterogeneous graph.

3.5. Metapath-Type Division

The metapath describes the relationship between two transaction entities and the shared wallet address. In this framework, depending on the division of wallet address types in Section 3.4, we use Algorithm 2 to define three metapath types to distinguish the subtle differences in Bitcoin transactions in detail, as shown in Figure 4.

As can be seen from the directed relationship in Figure 4, although Bitcoin transaction nodes are all connected to wallet addresses, different connection directions express different meanings. Algorithm 2 describes the metapath-type division method; the pseudocode is shown below.

Algorithm 2 Metapath Type Division**Input:** The Bitcoin transaction node data:

Txnode_List,

Three types of Bitcoin wallet address node:

Add_A, Add_B, Add_C

Output: Metapath of various types:

Metapathlist_A, Metapathlist_B, Metapathlist_C

```

1: function GET_INFO(tx)
  //using API https://chain.api.btc.com/v3/tx/{txhash} (accessed on 26 July 2023)
  to get jsonfile
2:   return tx.jsonfile
3: end function
4: for each out_edge  $\in$  Out_Edges do
5:   TransInfo=GET_INFO(tx)
6:   for each add_in  $\in$  TransInfo.InputAddress do
7:     add = add_in;
8:     In_Edges.append([add,tx])
9:   end for;
10:  for each add_out  $\in$  TransInfo.OutputAddress do
11:    add = add_out
12:    Out_Edges.append([tx,add])
13:  end for
14: end for
15: for each out_edge  $\in$  Out_Edges do
16:   for each in_edge  $\in$  In_Edges do
17:    if out_edge.add==in_edge.add then
18:      addnode = out_edge.add
19:      tx1_node = out_edge.tx
20:      tx2_node = in_edge.tx
21:      if addnode  $\in$  Add_A then
22:        Metapathlist_A.append([tx1_node,addnode,tx2_node])
23:      end if
24:      if addnode  $\in$  Add_B then
25:        Metapathlist_B.append([tx1_node,addnode,tx2_node])
26:      end if
27:      if addnode  $\in$  Add_C then
28:        Metapathlist_C.append([tx1_node,addnode,tx2_node])
29:      end if
30:    end if
31:  end for
32: end for

```

In all the edges of the Bitcoin transaction entities and the Bitcoin wallet address, we define the edge of the connection direction from the Bitcoin wallet address node to the Bitcoin transaction entity node as the In_Edge (line 8), inversely, as the Out_Edge (line 12). Then, we define the metapath type by classifying the combinatorial cases of In_Edge and Out_Edge (lines 15 to 32).

From the perspective of the Bitcoin transaction, in metapath A, both Tx_1 and Tx_2 have output to the common wallet address, which is more consistent with the situation of the centralized transfer of transactions (lines 21, 22). In metapath B, Tx_1 and Tx_2 show obvious one-way flow characteristics (lines 24, 25). In metapath C, both Tx_1 and Tx_2 have input from the common wallet address, which is more consistent with decentralized transaction transfer-out scenarios (lines 27, 28). Therefore, through the metapath setting, different means of the transaction can be reflected, aiming to input more characteristics of Bitcoin transactions into the model.

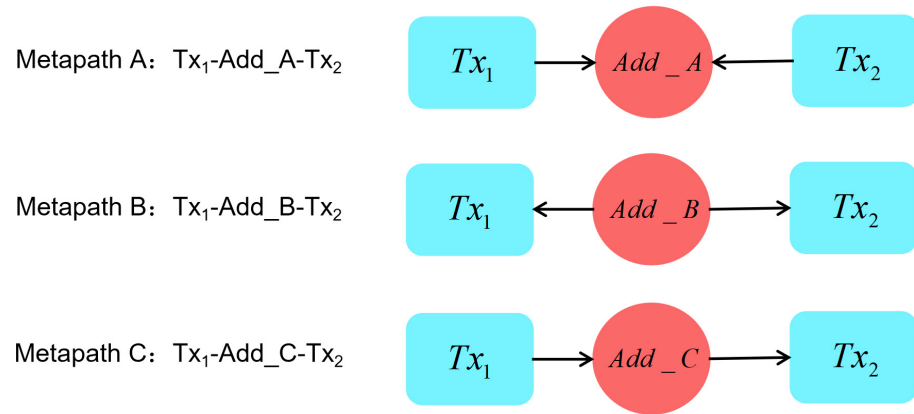


Figure 4. Metapath types in the Bitcoin heterogeneous transaction graph.

4. Methodology

MAGNN's idea is to input the encoded vectors into the graph neural network by designing the feature extraction and aggregation mode of the metapath so that the model can learn the embedded expression of the target node on the heterogeneous graph. MAGNN, which involves the handling of vectors, is manifested in intra-metapath aggregation and inter-metapath aggregation.

4.1. Intra-Metapath Aggregation

Given a metapath P , the aggregation encoder learns the embedding representation of the target node. Let $P(Tx_v, Add, Tx_u)$ be one of the metapath instances of the target node Tx_v , and convert all node features along the metapath instances into a single vector through the encoder f_θ .

$$h_{P(Tx_v, Add, Tx_u)} = f_\theta(P(Tx_v, Add, Tx_u)) = f_\theta\{h_{Tx_v}, h_{Add}, h_{Tx_u}\} \quad (1)$$

After the metapath instances are encoded as vector representations, the multi-head attention mechanism [26] is used to carry out a weighted summation of metapath instances in P that relate to the target node Tx_v . The core idea is that different metapaths represent target nodes in various degrees. We can model this by learning a normalized importance weight for each metapath and then a weighted sum for all instances:

$$h_{Tx_v}^P = \frac{1}{K} \sum_{k=1}^K \left(\sum_{Tx_u \in N_{Tx_v}^P} \left[\alpha_{Tx_v Tx_u}^P \right]_k \cdot h_{P(Tx_v, Add, Tx_u)} \right) \quad (2)$$

$\left[\alpha_{Tx_v Tx_u}^P \right]_k$ is the normalization importance of the metapath instance to the node Tx_v at the K^{th} attention head.

4.2. Metapath Aggregation Encoder

In the metapath proposed in this paper, the length of the metapath is at fixed length 3, and the transaction entity is a symmetric relationship centered on the wallet address. Therefore, we designed an encoder based on the idea of TransE in the knowledge graph to extract the features of the metapath.

$$\begin{aligned} t &= h'_{Add} \\ o_1 &= h'_{Tx_v} + t \odot r_1 \\ o_2 &= h'_{Tx_u} + t \odot r_2 \\ h_{P(Tx_v, Add, Tx_u)} &= o_1 \oplus t \oplus o_2 \end{aligned} \quad (3)$$

We list all possible edge types in the metapath, namely $Tx - Add_A$, $Tx - Add_B$, $Tx - Add_C$, $Add_A - Tx$, $Add_B - Tx$, and $Add_C - Tx$. For different kinds of edges, we initialize different relational vectors r_i and abstract the eigenvectors of transaction nodes into h , and the eigenvectors of wallet address nodes into t in the relation triplet (h, r, t) .

The idea of the TransE algorithm is that a correct triplet of embeddings will meet, $h + r = t$, so we will migrate this idea to the metapath node aggregation, aiming to learn better embedding characteristics through the association of the Bitcoin transaction entity and wallet address. The encoder we designed is shown in Figure 5.

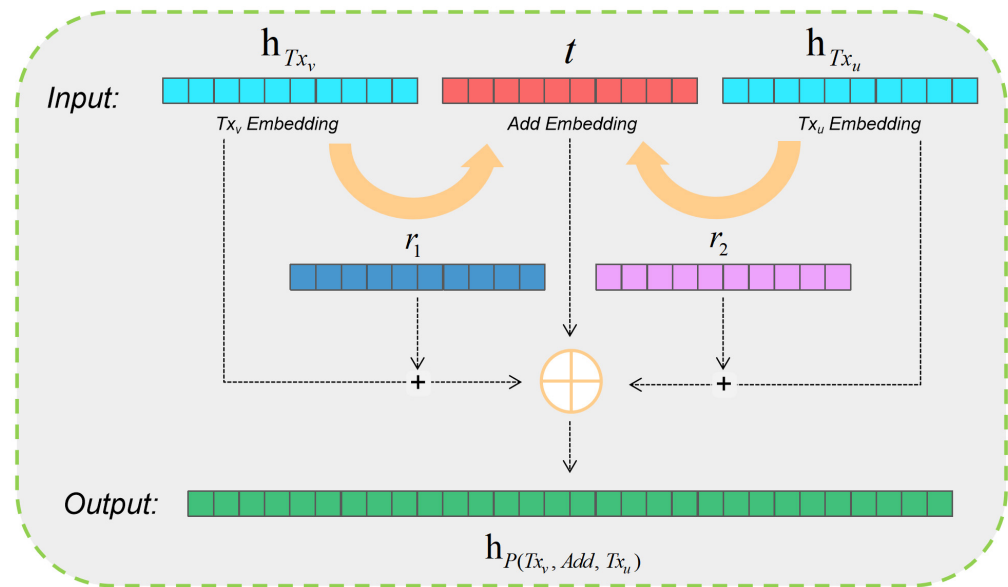


Figure 5. Metapath aggregation encoder based on the TransE idea.

4.3. Inter-Metapath Aggregation

After aggregating the transaction node and wallet address node of each metapath, the inter-metapath aggregation layer is used to express the embedded characteristics of the target node. For a node Tx_v , the vector set is $\{h_{Tx_v}^{P_1}, h_{Tx_v}^{P_2}, h_{Tx_v}^{P_3} \dots h_{Tx_v}^{P_S}\}$, where S is the number of metapaths of the target transaction node. Next, we average the metapath's specific node vectors after the transformation of all transaction nodes:

$$s_{P_i} = \frac{1}{|V_{Tx}|} \sum \tanh(M_{Tx} \cdot h_{Tx}^{P_i} + b_{Tx}) \quad (4)$$

where M_{Tx} and b_{Tx} are learnable parameters. Then, we use the attention mechanism to fuse each transaction metapath $P_i \in \mathcal{P}_{Tx}$, as follows:

$$\begin{aligned} e_{P_i} &= q_{Tx_v}^T \cdot s_{P_i} \\ \beta_{P_i} &= \frac{\exp(e_{P_i})}{\sum_{P \in \mathcal{P}_{Tx_v}} \exp(e_P)} \\ h_{Tx_v}^{xv} &= \sum_{P \in \mathcal{P}_{Tx_v}} \beta_P \cdot h_{Tx_v}^P \end{aligned} \quad (5)$$

To summarize, the extraction and coding processes of the metapath of Bitcoin transactions are shown in Figure 6.

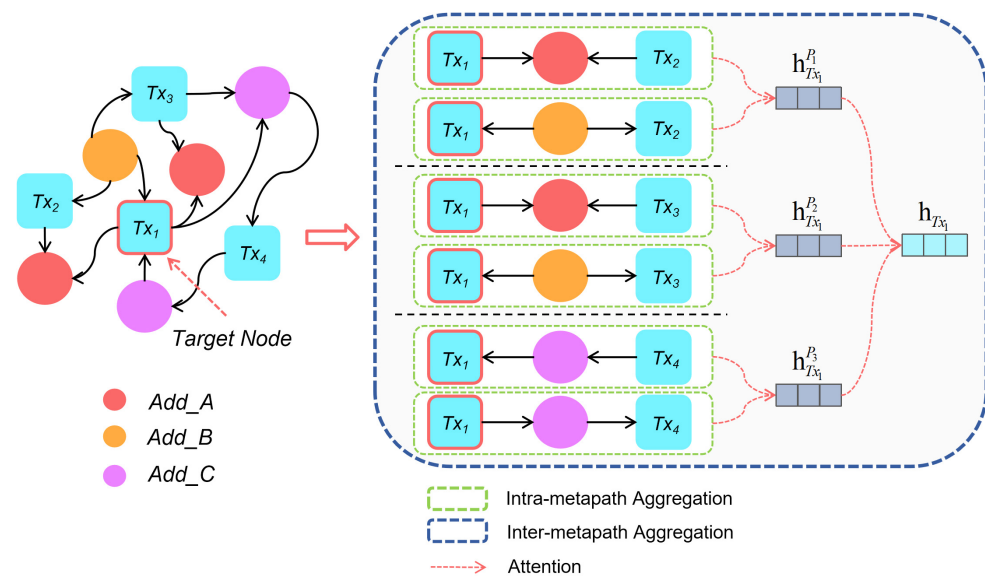


Figure 6. Metapath extraction and coding process.

5. Experiments and Discussions

In this section, we investigate the effectiveness of the proposed method on the Bitcoin anti-money laundering task, intending to answer the following questions.

Q1: Which model shows better classification results on HBTBD?

Q2: Which is more important, graph embedding features or attribute features?

Q3: How do the training data sizes impact model performance?

Q4: How well does the model generalize and maintain robustness on an imbalanced dataset?

5.1. Dataset

HBTBD is a heterogeneous graph of the Bitcoin transaction dataset we constructed using the method in Section 3.2, as shown in Table 2. The dataset contains 46,045 transaction entities (licit/illicit) associated with 319,311 Bitcoin wallet address signatures and metapath information. Attribute features can be divided into two parts by node type.

Table 2. Statistics of datasets.

Dataset	Node	Attributes	Metapath
HBTBD	Tx:29699	Tx(165 items):	Tx-Add_A-Tx:229916
	Add_A:68560		Tx-Add_B-Tx:284066
	Add_B:113121		Tx-Add_C-Tx:248586
	Add_C:16796		
	Tx:16346	Add(8 items):	Tx-Add_A-Tx:123211
	Add_A:59441		Tx-Add_B-Tx:148005
Test data	Add_B:49162		Tx-Add_C-Tx:126777
	Add_C:12231		
Total	Tx:46045	Add:319311	Metapath:1160561

Bitcoin transaction features: Each node has 166 associated features. The first 94 features represent local information about the transaction, including the timestep, the number of inputs/outputs, transaction fee, output volume, aggregated figures, etc. The remaining 72 features, called aggregated features, are obtained by aggregating transaction information one-hop backward/forward from the center node giving the maximum, minimum, and standard deviations, and correlation coefficients of the neighbor transactions for the same

information data. The first 94 features represent local information about the transaction, including the number of inputs/outputs, transaction fee, output volume, etc. The remaining 72 features represent structural features obtained by aggregating transaction information from the target node and its neighbor nodes, the maximum, minimum, standard deviation, etc. Wallet address features: Each node has eight associated features representing the specific Bitcoin wallet address information, including input/output, balance, number of transactions participated, pending transactions, pending income, pending expenditure, and unspent transactions.

The computer configuration used in the experiment is shown in Table 3. At the same time, we made the experimental code public on GitHub (<https://github.com/Tsunami-Song/Bitcoin-Laundering-Identification>, accessed on 26 July 2023).

Table 3. Computer configuration.

Hardware/Software	Configuration
CPU	Intel(R) Core(TM) i5-12450H 2.00 GHz
RAM	16 GB
GPU	NVIDIA GeForce RTX 3050 Laptop
Operating system	Windows 11 Professional
Python	3.7.13

5.2. Experimental Setup

Our experimental work aims to learn the embedded expressions of transaction nodes of different models. The detection of illicit transactions of Bitcoin is of great importance when selecting transaction features. Random forest can better capture more valuable features among numerous features. Therefore, our practice first learns the embedded features on the test set and then uses the traditional machine learning model, random forest, for classification tasks, as shown in Figure 7.

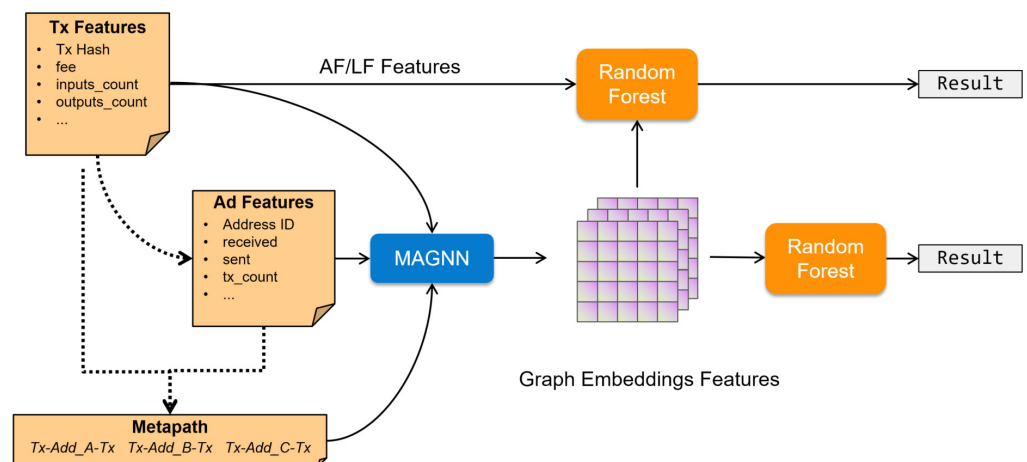


Figure 7. The experimental framework of a Bitcoin anti-money laundering task.

Considering the timestep factor in the dataset and the baseline setting by Weber et al. [10], we take 1–30 timesteps as the training set, 30–34 timesteps as the verification set, and 35–49 timesteps as the test set. Following the experimental ideas by Weber et al., we ran experiments on our experimental framework. AF and LF refer to full features and local features, respectively. Through the training of the first 35 timesteps, we, respectively, obtained the embedding vectors of GCN, GAT, EvolveGCN, HAN, and MAGNN with the same dimensions as the next 35–49 timesteps. Among them, the GCN, GAT, and EvolveGCN models were the experiments conducted by converting the graph into homogenous format.

- (1) GCN: A homogeneous GNN that performs graph convolutions in the Fourier domain.
- (2) GAT: A homogeneous GNN with spatial convolutions and attention mechanism.
- (3) EvolveGCN: A homogeneous GNN that uses RNN to evolve GCN parameters to capture the dynamic information of graph sequences.
- (4) HAN: A heterogeneous GNN that learns metapath-specific node embeddings and uses attention mechanism to merge them into node representations.
- (5) MAGNN: A heterogeneous GNN that learns metapath-specific node embeddings and uses an encoder for inter-path information aggregation into node representations.

The embedding representation obtained through GNN is spliced with the original AF or LF attribute features, and the classification is finally achieved by random forest and SVM.

It should be noted here that to avoid randomness, we set the `random_state` of 0 on the initialization of the `train_test_split` method. The classifier was constructed using the random forest and SVM classifier methods of Python's Sklearn [27] module. The parameters were set as `random_state = 0`, `n_estimators = 50`, `max_features = 50`, `n_jobs = -1` in random forest and `kernel = 'linear'`, `C = 1`, `gamma = 'auto'`, `probability = True` in SVM.

5.3. Discussion and Result

5.3.1. Performance Comparison (Q1)

Based on the collected dataset and experimental setup, we tested the performance of several state-of-the-art graph neural network methods to verify their effectiveness on the HBTBD dataset using random forest and SVM classifiers, and the corresponding precision, recall, and F1 scores are reported in Table 4, respectively. We also tested the classifier results without the graph embedding features. We have three findings, as follows:

Table 4. Classification results with original attribute features.

Method	Illicit		
	Precision	Recall	F1
Random Forest (LF + GCN)	0.812	0.584	0.679
Random Forest (AF + GCN)	0.841	0.603	0.702
Random Forest (LF + GAT)	0.720	0.587	0.647
Random Forest (AF + GAT)	0.708	0.580	0.638
Random Forest (LF + EvolveGCN)	0.824	0.582	0.679
Random Forest (AF + EvolveGCN)	0.853	0.611	0.712
Random Forest (LF + HAN)	0.809	0.613	0.697
Random Forest (AF + HAN)	0.805	0.615	0.697
Random Forest (LF + MAGNN)	0.907	0.605	0.726
Random Forest (AF + MAGNN)	0.922	0.617	0.739
SVM (LF + GCN)	0.778	0.602	0.679
SVM (AF + GCN)	0.820	0.618	0.705
SVM (LF + GAT)	0.706	0.622	0.661
SVM (AF + GAT)	0.683	0.628	0.654
SVM (LF + EvolveGCN)	0.831	0.615	0.707
SVM (AF + EvolveGCN)	0.850	0.624	0.720
SVM (LF + HAN)	0.833	0.630	0.717
SVM (AF + HAN)	0.859	0.626	0.724
SVM (LF + MAGNN)	0.901	0.620	0.735
SVM (AF + MAGNN)	0.911	0.637	0.750
Random Forest(LF)	0.865	0.586	0.699
Random Forest(AF)	0.789	0.609	0.687
SVM(LF)	0.831	0.597	0.695
SVM(AF)	0.849	0.602	0.704

First, the feature embedding of GNN strengthens the model performance to some extent. We can see that the precision, recall, and F1 scores of most models increase after adding the graph embedding feature, so it can be shown that the graph structure information is effective in the Bitcoin anti-money laundering task.

Second, We found that MAGNN is better than the other models under our designed metapath encoder; its precision reached the highest of 0.922 over EvolveGCN with about 7% better accuracy under random forest, and its recall also reached the highest of 0.637 by 1% more than GAT under SVM.

Third, we found that random forest and SVM also have different metrics. Random forest is better on precision, but SVM is better on recall. The different classification strategies may cause this.

5.3.2. Feature Importance Evaluation (Q2)

To better measure the importance of the embedding features obtained from graph representation learning versus the original features, we explore from the classifier perspective. We evaluate this from two perspectives: the random forest classifier for the tree model and the SVM classifier for the linear model.

We use the `feature_importances_` method in random forest and the `svm.coef_` method in SVM to determine the importance of all input classifier features; the higher the feature score, the more important the feature is.

The feature scores are plotted as bar graphs, distinguishing the two feature classes by color, i.e., blue for the original AF features and red for the graph embedding features obtained through graphical representation learning. We selected the embedded features of the GCN, EvolveGCN, and MAGNN models to join with the original AF features for the experiments, as shown in Figure 8.

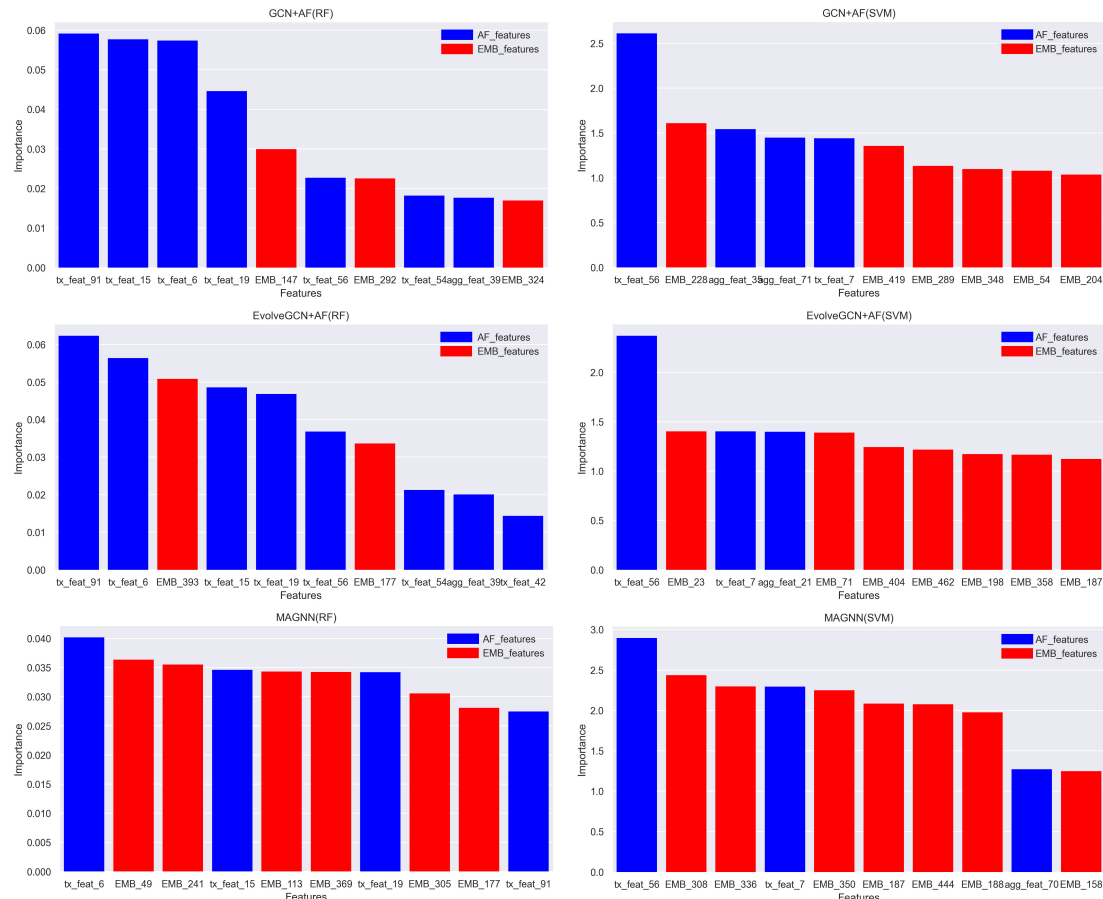


Figure 8. Importance of each feature under the random forest classifier (top 10).

5.3.3. Effect of Training Data Size (Q3)

We compared GNN models with different training data sizes to detect the model performance and robustness under label imbalance datasets. To exclude interference from the original features, we discarded AF and LF features. We used SVM as the classifier of experiments to evaluate precision, recall, and F1 scores for each model under different training data sizes. The results of this experiment are shown in Figure 9.

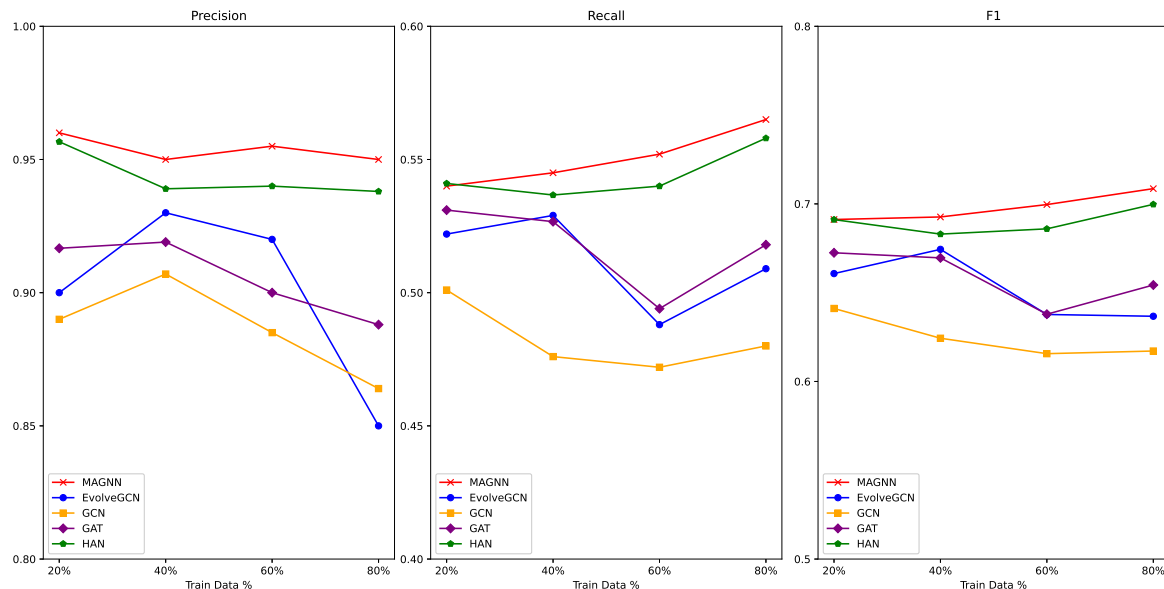


Figure 9. Graph embedding model classification results.

As we can see from Figure 9, MAGNN has undeniable advantages. For the most part, MAGNN's metrics (precision, recall, F1) remain high and stable.

We found that after increasing the training data size, the MAGNN and HAN model slightly improved in recall and F1 scores. At the same time, the precision of GCN, GAT, and EvolveGCN continuously decreased with the increase of training data, and the change range of each score was large and unstable. Therefore, we believe that as heterogeneous graphs, MAGNN and HAN are robust and have the advantage of detecting small numbers of samples in unbalanced datasets.

5.3.4. Model Generalization Ability on Imbalanced Datasets (Q4)

To test the generalization ability and robustness of the model on the unbalanced dataset and prevent the interference of label proportion inconsistent factors at different timesteps, we shuffled the data at all timesteps, reset the training data and test data, and evaluated the ROC curve and the AUC score.

The ROC curves show the trade-off between sensitivity, recall ratio, and specificity. ROC curves and the AUC index are more suitable for objectively evaluating unbalanced label classification tasks. We plotted the ROC curve and calculated the AUC of the SVM classifier, as shown in Figure 10.

According to the ROC curves and AUC score, the embedding features obtained by MAGNN outperforms the other models in all training data sizes. It can, therefore, be speculated that MAGNN has good model generalization ability and robustness on unbalanced datasets.

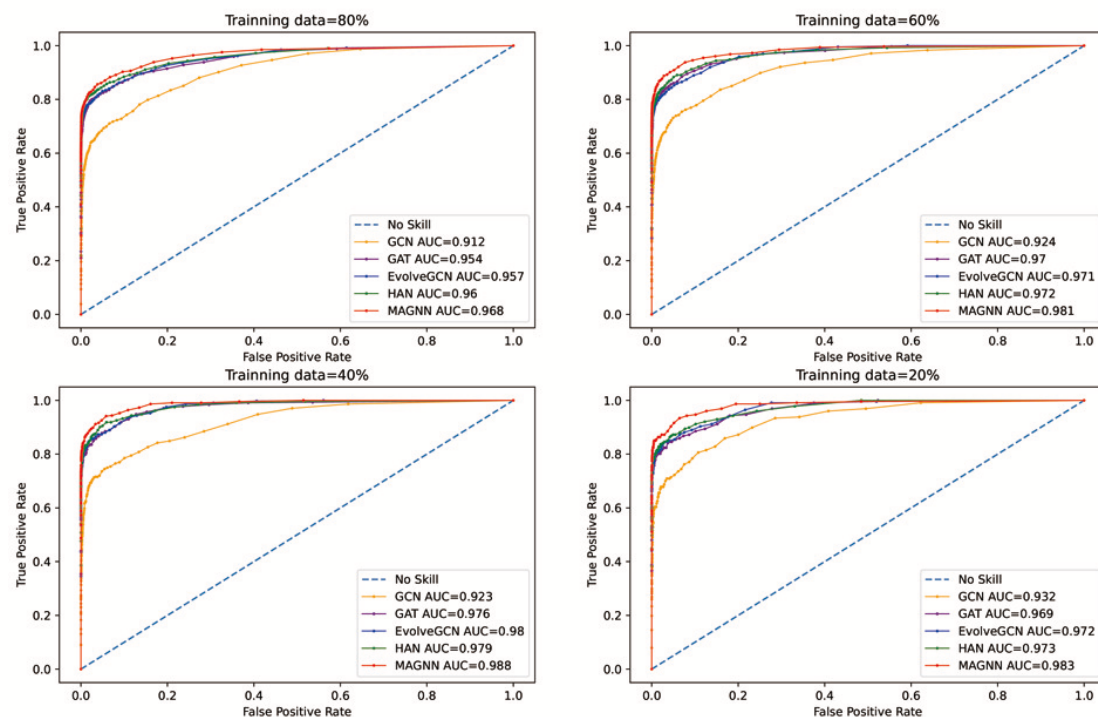


Figure 10. ROC curve and AUC calculation results under different training sizes.

6. Conclusions

In the Bitcoin anti-money laundering task, the method of constructing a heterogeneous graph transaction network has mined the characteristics of Bitcoin transactions, to a certain extent. The introduction of wallet address nodes helps to enrich the metapath information. Through the metapath encoder, we captured the relationship between the transaction entity and wallet address in the transaction, and superimposed it into the embedded expression of the transaction node. With the MAGNN mode, a better embedding expression was obtained, and a relatively complete Bitcoin anti-money laundering construction and detection framework was realized.

The improvement of the classification task from homogeneous to heterogeneous graphs is inseparable from the continuous enrichment of attribute information and enhancement of methods. We believe that adding features associated with transactions can further enhance the accuracy of Bitcoin anti-money laundering classification tasks. For example, more statistical and subgraph features can be enriched through data collection. At the same time, the encoding mode and path aggregation mode of the input features of the heterogeneous graph embedding model can be optimized to improve the effectiveness of the embedded expression.

Author Contributions: Data curation, J.S.; formal analysis, J.S.; investigation, J.S. and Y.G.; writing—original draft preparation, J.S.; writing—review and editing, J.S. and Y.G.; methodology, J.S.; supervision, Y.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Double First-Class Innovation Research Project for the People's Public Security University of China (no. 2023SYL07).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The HBTBD (heterogeneous Bitcoin transaction behavior dataset) is available at: <https://www.kaggle.com/datasets/songjialin/hbtbd-for-aml>, accessed on 26 July 2023.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Fletcher, E.; Larkin, C.; Corbet, S. Countering money laundering and terrorist financing: A case for bitcoin regulation. *Res. Int. Bus. Financ.* **2021**, *56*, 101387. [CrossRef]
2. Jiang, M.; Liu, G.; Su, Y.; Wu, X. GCN-SL: Graph Convolutional Networks with Structure Learning for Graphs under Heterophily. *arXiv* **2021**, arXiv:2105.13795.
3. Liu, Y.; Ao, X.; Qin, Z.; Chi, J.; Feng, J.; Yang, H.; He, Q. Pick and choose: A GNN-based imbalanced learning approach for fraud detection. In Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; pp. 3168–3177.
4. Claudio Bellei, A. The Elliptic Data Set: Opening up Machine Learning on the blockchain. Medium (Aug.2019). 2019. Available online: <https://medium.com/elliptic/the-elliptic-data-set-opening-up-machine-learning-on-the-blockchain-e0a343d99a14> (accessed on 26 July 2023) .
5. Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; Yakhnenko, O. Translating embeddings for modeling multi-relational data. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 2787–2795.
6. Fu, X.; Zhang, J.; Meng, Z.; King, I. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In Proceedings of the Web Conference 2020, Taipei, Taiwan, 20–24 April 2020; pp. 2331–2341.
7. Platt, J. Sequential Minimal Optimization: A Fast Algorithm for Training. 1998. Available online: <https://www.microsoft.com/en-us/research/publication/sequential-minimal-optimization-a-fast-algorithm-for-training-support-vector-machines/> (accessed on 26 July 2023).
8. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
9. Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
10. Weber, M.; Domeniconi, G.; Chen, J.; Weidele, D.K.I.; Bellei, C.; Robinson, T.; Leiserson, C.E. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv* **2019**, arXiv:1908.02591.
11. Bruna, J.; Zaremba, W.; Szlam, A.; LeCun, Y. Spectral networks and locally connected networks on graphs. *arXiv* **2013**, arXiv:1312.6203.
12. Pareja, A.; Domeniconi, G.; Chen, J.; Ma, T.; Suzumura, T.; Kanezashi, H.; Kaler, T.; Schardl, T.; Leiserson, C. Evolvegc: Evolving graph convolutional networks for dynamic graphs. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 5363–5370.
13. Shi, X.; Chen, Z.; Wang, H.; Yeung, D.Y.; Wong, W.K.; Woo, W.C. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 802–810.
14. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.
15. Hu, Y.; Seneviratne, S.; Thilakarathna, K.; Fukuda, K.; Seneviratne, A. Characterizing and detecting money laundering activities on the bitcoin network. *arXiv* **2019**, arXiv:1912.12060.
16. Perozzi, B.; Al-Rfou, R.; Skiena, S. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 701–710.
17. Grover, A.; Leskovec, J. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 855–864.
18. Lorenz, J.; Silva, M.I.; Aparício, D.; Ascensão, J.T.; Bizarro, P. 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, New York, NY, USA, 15–16 October 2020; pp. 1–8.
19. Oliveira, C.; Torres, J.; Silva, M.I.; Aparício, D.; Ascensão, J.T.; Bizarro, P. GuiltyWalker: Distance to illicit nodes in the Bitcoin network. *arXiv* **2021**, arXiv:2102.05373.
20. Jensen, R.I.T.; Iosifidis, A. Qualifying and raising anti-money laundering alarms with deep learning. *Expert Syst. Appl.* **2023**, *214*, 119037. [CrossRef]
21. Yang, G.; Liu, X.; Li, B. Anti-money Laundering Supervision by Intelligent Algorithm. *Comput. Secur.* **2023**, *132*, 103344. [CrossRef]
22. Zhao, K.; Dong, G.; Bian, D. Detection of Illegal Transactions of Cryptocurrency Based on Mutual Information. *Electronics* **2023**, *12*, 1542. [CrossRef]
23. Xiang, Y.; Ren, W.; Gao, H.; Bao, D.; Lei, Y.; Li, T.; Yang, Q.; Liu, W.; Zhu, T.; Choo, K.K.R. BABD: A Bitcoin Address Behavior Dataset for Address Behavior Pattern Analysis. *arXiv* **2022**, arXiv:2204.05746.
24. Alexander. De-Anonymization of Elliptic Dataset Transactions. 2019. Available online: <https://habr.com/ru/post/479178/> (accessed on 26 July 2023).
25. Sun, Y.; Han, J. Mining heterogeneous information networks: A structural analysis approach. *ACM Sigkdd Explor. Newsl.* **2013**, *14*, 20–28. [CrossRef]

26. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
27. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.