Barcelona School of Economics

MSc Data Science Methodology

Risk detection in cryptocurrency markets: meeting the needs of traditional finance

Iván Aguilar
Rebecca Jones
Gian-Piero Lovicu

May 2022

As the popularity of cryptocurrencies increases, traditional financial institutions are beginning to integrate them into their payment systems. However, to do so, these institutions must comply with anti-money laundering (AML) regulations that take reasonable steps to avoid serving or transacting with counterparties that are at high risk of involvement in criminal activities. We consider a financial institution's key requirements for a risk detection model to address AML regulations: it should prioritise a high recall, be scalable to large amounts of data and provide the capability to explain why a counterparty is high risk. These are areas which previous literature do not address. In light of these requirements, we propose a novel way to organise Bitcoin data to allow the counterparties (addresses) to a transaction form the input into a risk detection model, instead of the transaction itself (which could potentially involve many counterparties in Bitcoin). We test different Graph Neural Network (GNN) models as well as some baseline supervised classifiers. We find that a Graph Attention Network (GAT) using our novel address-level data achieves the best recall: capturing 83 per cent of high risk addresses by volume and 96 per cent by value transacted. This is an improvement on the results achieved in previous literature. We were unable to satisfactorily address the scalability or explainability requirements, though we discuss why and provide suggestions for future work to explore these issues further.

1 Introduction

Financial institutions are beginning to integrate their digital payments systems with cryptocurrency markets, for example by processing transactions between normal (fiat) money and cryptocurrency for clients. Bitcoin was the first cryptocurrency to emerge in 2009, and its lack of necessity for a trusted intermediary to execute transactions was revolutionary. At present, thousands of different cryptocurrencies are in circulation and are increasingly used as a means of payment and investment. The two most popular cryptocurrencies today are Bitcoin and Ethereum and make up around two-thirds of total cryptocurrency market capitalisation (CoinMarketCap, 2022).

To comply with anti-money laundering (AML) regulations, financial institutions must know from whom they are accepting funds to monitor and control the risk of transacting with counterparties that are involved in criminal activities. AML regulation includes dealings in cryptocurrencies. Risky counterparties range from crypto-finance institutions who do not perform background checks on their customers (called Know Your Customer or KYC protocols) and so can provide a haven for criminal activity, to transacting with the criminals directly.

In identifying these 'high risk' counterparties, financial institutions have several requirements (and accompanying challenges) for a risk detection model:

- The regulatory and reputational costs of non-compliance with AML rules are high, so financial institutions are naturally conservative when it comes to identifying high risk activity. In other words, the cost of a false negative is much higher than a false positive. In practice, this means that financial institutions are willing to trade-off some precision in their risk detection models to ensure their **recall is high**.
- Financial transactions produce a continuous flow of data that cumulates very quickly. Therefore, risk detection models must be **scalable to large amounts of data**.
- Internal processes usually require a risk detection model to offer an explanation, if not proof, as to why a transaction is high risk. This is because declining to transact with a potential counterparty is not costless - the opportunity cost is revenue. Therefore, some **explainability** is a desirable feature of a risk detection model. Automating explainability is pertinent as compliance costs continue to grow in financial institutions following the global financial crisis (DeloitteUS, 2021).

Cryptocurrencies provide a need for new approaches to risk detection for financial institutions. Cryptocurrency data are anonymous since a participant does not have to disclose their identity to transact on the blockchain. However, since the blockchain is public all transaction data are freely available to download. Therefore risk detection models can make use of the full history of transaction data, which are naturally structured as a graph. It transpires that even with anonymity, transaction patterns make it possible to identify who is transacting with only a little prior information (Agarwal, 2020).

In this thesis we use a labelled dataset of Bitcoin transactions to test different approaches to detecting high risk activity. Our data is based on the stylised Elliptic dataset (Weber et al., 2019), which contains around 200,000 Bitcoin transactions from 2016 to 2017, a portion of which are labelled illicit and licit (which we relabel to high and low risk). We extract the full set of raw data underlying the transactions directly from the blockchain. We then combine the Elliptic data with another set of labels identifying high and low risk counterparties (obtained from a Barcelona-based crypto security firm, Clovr Labs). Finally, we use this augmented data to construct two types of datasets: a transaction-level dataset — that follows the approach of the rest of the literature — and a novel address-level dataset. The address-level dataset has the advantage of being able to identify risky activity at the counterparty level. Both datasets are highly imbalanced away from high risk observations, which is a typical problem faced in risk detection exercises in finance (i.e. the vast majority of transactions are low risk). We acknowledge that these data are somewhat dated, given Bitcoin is significantly more mainstream in 2022 than in 2017, and that this could affect the external validity of our results.

We try several classifier models to detect high risk activity. We focus on achieving a high recall, though we attempt to balance this against the cost of a lower precision. As a baseline we use traditional supervised classifiers: a linear logistic regression classifier and a non-linear random forest classifier. These baseline models have the advantage of some explainability and, in the case of random forest, perform well for most tasks. However, they also treat the input data as identically independently distributed (iid) which discards the natural graph structure of the data. Therefore, we also train some neural network models that utilise the graph structure of the data to generate euclidean embeddings. Our preferred model is a graph attention network (GAT), which augments the baseline graph convolutional network (GCN) by 'learning' how important the neighbours of a particular graph node are in learning its embedding. In all of our models, we correct for class imbalance by undersampling low risk observations from our dataset.

Overall our best performing model is the GAT applied to the address-level dataset. This compares favourably to our baseline random forest model and to the highest recall we observed in the literature using the Elliptic dataset (Table 1).[1] In addition, when we measure these metrics by the value of transactions, the recall of our GAT model improves substantially to capture almost all of the value of the high risk transactions. Its precision also improves substantially. In contrast, our baseline random forest model performs much worse by this measure. The GAT is an attractive candidate for a risk detection model because it can go some way to addressing explainability, though this is not something we were able to explore in this work. Our transaction-based models perform worse than the literature, though none of the papers we looked at managed to achieve an acceptably high recall score on transaction-based data.

	Recall	Recall(by value)	Precision	Precision (by value)
GAT (candidate)	0.83	0.96	0.26	0.74
RF (baseline)	0.79	0.59	0.27	0.58
Arcos-Diaz(lit)	0.72	-	0.92	-

Table 1: Summary of key results using address-level data (test data)

Previous literature also uses cryptocurrency data - mostly based on Bitcoin and Ethereum — to train classification models to detect fraud and/or other illicit activities (Weber et al., 2019), (Wu et al., 2020), (Alarab et al., 2020). The majority of the literature focuses on shallow and deep embedding methods using graph data to achieve this aim. Many papers are quite successful at predicting fraud and/or illicit activity from transaction patterns on the blockchain and most have a particular focus on phishing, for which there is quality labelled data available (especially for Ethereum). However, this literature does not consider the specific needs of traditional financial institutions. This is somewhat expected, given traditional finance is only a new entrant in the cryptocurrency space.

This thesis differentiates itself from the literature by exploring the specific needs of financial institutions outlined above. We also introduce a novel address-based way to organise data from the Bitcoin blockchain based on the counterparties forming a transaction. This departs from the rest of the literature on Bitcoin, which focuses on the transactions themselves, which may consist of (potentially) many counterparties. Overall, we are successful in achieving the first requirement, proposing a model that outperforms the literature on the key recall metric, without trading off too much of the model's precision. However, scalability and explainability are more challenging propositions, which we discuss in the analysis section of the thesis. We also propose some ideas on how further work can address these challenges.

In this thesis we have used a smaller stylised dataset that is not reflective of the large scale of data a real financial institution would have to deal with. As a potential solution, we propose how a financial institution may combine elements of our thesis to construct a two-step 'pipeline' that scales to large amounts of data. Specifically, we note that our candidate model performs well on balanced data (across all metrics) and could form the second step of the pipeline. Then, the missing first step of the pipeline is a method to confidently discard (unlabelled) observations that are likely low risk. This would help to balance the data passed to the address-level model and allow it to scale to processing much larger amounts

[1] The results in Arcos-Diaz (2019) are based on a model trained on transaction-level data.

of data. Our candidate to achieve this was the transaction-based model, though as we discuss we were unable to improve its performance from existing papers.

As a first approach to explainability, we anecdotally explore some summary statistics based on the predictions made by our model. Through this, we observe that high risk addresses tend to transact in lower amounts and with fewer counterpart addresses. This is interesting, but does not really reveal anything about the drivers underlying the predictions of our model. We acknowledge that this analysis is simplistic and we suggest how this may be improved in future work.

The rest of this thesis is organised as follows. Sections 2 and 3 provides a background on a cryptocurrency blockchain and Bitcoin. Section 4 discusses the types of high risk activity using cryptocurrencies and Section 5 discusses similar literature in this area. Sections 6 and 7 outline our data and Section 8 our modelling framework. In Section 9 we present results and in Section 10 some analysis, limitations and recommendations for future work.

2 What is Bitcoin?

Bitcoin is a peer-to-peer form of electronic cash. Bitcoin relies on a distributed network, based on a technology called blockchain, to allow payments between individuals without the need for a trusted third-party (such as a financial institution) to verify the transaction, which is how traditional payments work. Understanding the blockchain technology underlying the Bitcoin payment system is crucial for understanding how Bitcoin transactions are recorded. This is especially since these data are different from how we usually conceptualise recording transactions in a traditional bank account.

A blockchain is a record of information made up of append only, timestamped data logs (the blocks of the chain) and a decentralised mechanism to achieve consensus between the participants in the system. The concept of blockchain precedes Bitcoin and was invented in the mid 1990's as a way to timestamp digital documents.[2] In the case of electronic payments, the append only logs of a blockchain are typically ledgers of transactions, similar to in traditional accounting. These form the 'blocks' that are chained together using a one-way cryptographic function (called a hash).[3] Once a block is added to the chain, it is time-stamped and quickly becomes immutable as new blocks are chained after it. This means that any change to an earlier block (e.g. by an attacker attempting to reverse a payment made previously, so they may spend the funds again) changes its cryptographic representation and therefore the cryptographic representation of all blocks after it. This invalidates the version of the blockchain with the altered record.

Bitcoin is a public, permissionless blockchain. This means any individual can become a user of Bitcoin and can transact with other users. Bitcoin uses digital signatures, a form of asymmetric cryptography, to secure transactions across the network. Each user in the network has a private and public key pair. The private key secures a user's Bitcoins, while the public key (transformed cryptographically into a more user-friendly Bitcoin address) can verify that a digitally-signed message originates from the paired private key. In essence, when a user wants to transact they digitally sign a transaction using their private key and broadcast it to the network . Other participants in the network can then easily verify, using the public key/bitcoin address, that the transaction indeed comes from the private key that controls the Bitcoins (Figure 1).

[2]Two computer scientists founded a company called Surety based on their technology, which provided digital notarisation of documents. The proof of notarisation was published every Sunday in the New York Times, which provided a timestamp and immutable record of the document verification. This means that physical weekly editions of the New York Times were the 'blocks' of the first blockchain (NewsBTC, 2018).

[3]A one-way (or asymmetric) cryptographic hash function encrypts an input into a fixed-length sequence of random alphanumeric characters (the hash). The asymmetry comes from the fact that it is infeasible to retrieve the original input from the hash, but costless to verify that the hash represents the purported input. These functions are powerful because an infinitesimally small change to the input will result in a completely new hash, rendering it impossible to make changes to the input without detection.

Figure 1: Digital signatures with private and public keys (Source: NIST)

The next step is to reach a consensus on which transactions are valid and render them immutable. Bitcoin achieves this consensus through a validation mechanism called proof-of-work (Back et al., 2002).[4] In proof-of-work, miners must search for a particular hash that summarises the transactions in that block.

Once the hash is found, pending transactions in the new block are chained with the hash to all previous blocks (Figure 2). In return, the miner receives newly minted Bitcoin as a reward (called the coinbase transaction) plus any transaction fees users include to incentivise the inclusion of their transaction in the block. The higher the transaction fee the user offers, the quicker the transaction is likely to be validated by a miner. Miners then begin working on the next block to append to the chain, to compete for the new coinbase reward on offer. The Bitcoin network has a block validated and added to the network on average every 10 minutes.

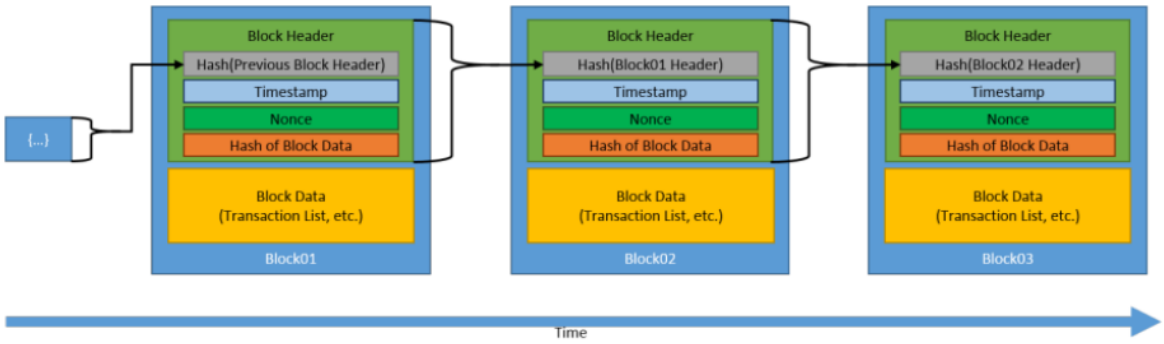

Figure 2: Example of a blockchain (Source: NIST)

3 How are Bitcoin transaction data recorded?

Bitcoin transactions follow the concept of 'Unspent Transaction Output (UTXO)'. This differs from how we think about transaction records in our regular bank account, where flows of funds in and out are netted off against each other to produce an account balance. We can view the transaction history at any time, but we summarise our account using the balance. Ledger systems of this nature are 'account-based'. The Ethereum blockchain is an example of an account-based ledger.

In contrast, Bitcoin follows a transaction-based ledger where the contents of an account (a 'wallet' with potentially many linked addresses) consist of all transactions where they received Bitcoins but have not yet spent them. These are called 'unspent outputs'. When a user wishes to send Bitcoin to another user(s), they collect enough unspent outputs to send. These are called 'inputs' into the transaction. The inputs are aggregated and form the new 'unspent outputs' of the receiving address(es). Usually, there is a transaction fee paid to the miner of the block, which is the difference between total inputs to total

[4] Many other consensus mechanisms exist for blockchains, for example the proof-of-stake mechanism which is popular since it is less computationally expensive than proof-of-work.

outputs. Since fees are paid per transaction, a user or users can pool their inputs to send to many receiving addresses, meaning that each transaction can potentially have many inputs and outputs. This can sometimes make it difficult to identify who is behind a transaction.

Figure 3 shows an example of a single transaction: user Alice sending 8 ZEN to user Bob. In state n, the users' wallets collect their unspent outputs. Alice uses an unspent output of 10 ZEN as input into a transaction where she sends 8 ZEN to Bob. The remaining 2 ZEN form another output which Alice sends to herself, as well as a small transaction fee to incentivise a block miner to include her transaction in a block. At state n+1 the set of unspent outputs has changed: Alice's 10 ZEN spent transaction (marked by the red 'S') no longer exists and is replaced by two unspent outputs: Bob's 8 ZEN and Alice's 1.999 ZEN (the miner would also receive an unspent output from the fee, but this is not shown).

Figure 3: A transaction in the UTXO model. Source: HorizenAcademy (2019)

This means that a given Bitcoin transaction will consist of 5 key data elements: the input addresses and amounts from each transaction, the output addresses and amounts from each transaction and the transaction summary which includes information such as the block time, total inputs and total outputs and whether the transaction is the coinbase reward (in which case it has no inputs) (Figure 4).

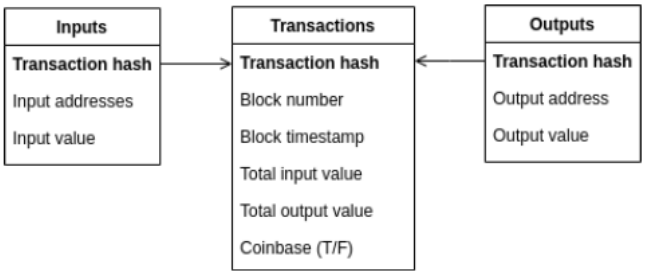

Figure 4: Bitcoin transactions - relational data

4 High risk activity

Financial institutions that take deposits of fiat money or make investments on behalf of customers are required to abide by AML rules, including having KYC protocols in place. These exist to try and prevent criminals from easily being able to move or store funds acquired from criminal activities. The advent of Bitcoin provided a new way for criminals to move and hide the proceeds from crime. As financial institutions establish businesses in the cryptocurrency space, there is a need for them to identify when they are at risk of dealing with individuals that have obtained their cryptocurrency from unlawful activities.

Since the inception of Bitcoin it has become increasingly difficult to use cryptocurrencies to hide criminal activity (DoJ, 2020). Chainalysis estimates that, in 2021, around 0.15 per cent of blockchain transaction volume involved criminal activity, a record low relative to previous years, having reached an estimated high of 3.37 per cent in 2019, with the high percentage in this year mainly driven by a

particularly prevalent Ponzi scheme (Chainalysis, 2022). The same report also notes that the majority of criminal activity was related to scams, rather than more serious unlawful activities. Although the level of criminal activity has fallen due to improved fraud detection methods, we note the importance of categorising and identifying such activity, in order to understand and implement controls to prevent it. Criminals are constantly evolving their methods in order to avoid detection and learning more about how they operate may help us predict their future behaviours.

In our data set, we are interested in three broad types of high risk addresses. The first are cryptocurrency exchanges with little to no KYC requirements for their customers. Crypto exchanges are the main gateway for users to switch between cryptocurrency and fiat money, or between cryptocurrencies. Without checks on their customers, these exchanges can provide a haven for criminal actors to move funds from crypto into fiat.

The second type of high risk addresses are associated with scams, summarised in Vasek and Moore (2015). These are typically cryptocurrency services that fraudulently pose as genuine to steal victims' money. Scams can include exchanges that offer enticing exchange rates, but then flee with a user's funds once their account exceeds a certain balance; fake donation campaigns; gambling scams where it is unclear whether users will be paid winnings. These also include phishing scams, where a scammer creates a fake version of a trusted website, and entices users to input their private information. In the case of cryptocurrency, scammers can target the victim's private key to gain control of their funds. Phishing is one of the most common forms of crypto scams and is the focus of much of the literature on crypto fraud detection.

The final type of high risk addresses are those associated with more serious criminal activities. These are agents that use cryptocurrencies to store and transact the proceeds of their activities such as terrorism, drug running, the dark market and child sexual abuse materials. They also include ransomware operations, where actors steal data from legitimate companies and demand a ransom in crypto.

5 Literature Review

Identifying and managing transaction risks related to criminal activities is a growing area of literature in the cryptocurrency space. Currently, regulatory costs for financial institutions to incorporate cryptocurrencies in their payment systems are high. Although cryptocurrency blockchains are largely public, they are also designed to preserve the anonymity of their users and do not universally apply standard risk management protocols. Users can also employ a number of trading strategies on the blockchain to enhance their privacy. Improving transparency and reducing the regulatory costs are crucial for a wider adoption of cryptocurrencies by traditional financial institutions (Weber et al., 2019).

In general, work in this area has generally focused on detecting illicit behaviour on the Bitcoin and Ethereum networks. In the literature cited, we do not always distinguish whether papers focus on Bitcoin or Ethereum. In general, papers focused on Bitcoin model data at the transaction-level, while papers focused on Ethereum model data at the account-level. As far as we are aware, there are not yet any papers modelling Bitcoin at the address-level, which is a novelty of our approach.

Detecting illicit behaviour is usually framed as a classification or outlier detection task. Classification in most instances is binary (i.e. illicit vs licit), though some papers also attempt to distinguish between different types of illicit activity. Examples of classifiers used in the literature range from a simple fully connected linear layer at the end of a neural network to popular supervised algorithms such as random forest, support vector machines and various forms of boosting (Weber et al., 2019), (Ostapowicz and Żbikowski, 2020), (Arcos-Diaz, 2019),(Alarab et al., 2020),(Wu et al., 2020),(Tian et al., 2021).

Outlier detection assumes that illicit activity exhibits anomalous transaction patterns relative to all licit activity. Models such as the one-class SVM then detect whether certain transactions are 'abnormal' relative to broader training data (Wu et al., 2020),(Patel et al., 2020). A benefit of outlier detection over

classification is that it helps to assuage problems with class imbalance that are typical when detecting illegal activity in financial transactions data.

Cryptocurrency transaction data are naturally modelled as a graph, while most classification and outlier detection algorithms require euclidean data. Therefore, most models aim to generate euclidean embeddings from graph data. These embeddings are treated as feature vectors that measure the similarity between pairs of graph nodes or edges. These embeddings can be used to train a classifier or outlier detection method outlined above. Broadly speaking, there are two classes of methods to generate graph embeddings: random walks and graph neural networks (GNNs).

Random walk methods are a so-called 'shallow' graph embedding technique. For each node, the algorithm moves to adjacent (neighbouring) nodes for a fixed sequence length, using a stochastic process that may depend on the relationship between the nodes. This process is then repeated many times (Perozzi et al., 2014), (Grover and Leskovec, 2016). Several papers have designed random walk algorithms specific to cryptocurrency data. Wu et al. (2020) define an algorithm called Trans2Vec for account-based cryptocurrency data (e.g. Ethereum). Trans2Vec assumes that transactions for larger amounts or closer together in time should have a higher probability of appearing together on the same random walk. The authors apply Trans2Vec to detect phishing accounts on the Ethereum network and find that it outperforms the baseline random walk methods. Other algorithms were proposed by Hu et al. (2019), Lin et al. (2020a), Xia et al. (2021) and Wang et al. (2021), some of which (notably) respect the sequencing of transactions in the random walk.

Random walk methods are attractive for their simplicity. However they are very computationally expensive to train (even more so than GNNs). This is problematic given the size of cryptocurrency transaction networks. Indeed, all of the papers mentioned use small, highly stylised datasets to train their embeddings. These data are not reflective of the large data sets a workhorse model in a financial institution would need to process and few papers discuss the potential for model scalability. In addition, random walk methods embed graph structure but ignore information captured in node and edge features. One workaround is to concatenate node features with random walk embeddings, like the SigTran algorithm created by Poursafaei et al. (2021) for fraud detection in Bitcoin and Ethereum. However, this treats graph structure and node features as independent, which is not necessarily a valid assumption. This is one advantage of GNNs, which treat graph structure and node features as dependent.

A number of papers have proposed GNN architectures for detecting different types of illicit activity on the Bitcoin and Ethereum networks. Some architectures build off a baseline Graph Convolutional Networks (Alarab et al., 2020), (Tan et al., 2021), (Tang et al., 2022), (Lou et al., 2020), while others combine graph convolutions with more sophisticated GNN features. For example, Tian et al. (2021) use a Graph Attention Network to identify illicit transactions on the Bitcoin network, while Kanezashi et al. (2022) use a heterogenous GNN on the Ethereum network to differentiate between node types, such as exchanges, wallets and regular accounts. Another approach is to combine the embeddings output from a GNN with a classifier other than a fully connected linear layer, such as a random forest (Kolesnikova et al., 2021).

Similar to the models that use shallow embedding techniques, the GNNs in these papers are trained on highly stylised data sets and few discuss the possibility for scalability. Some papers have taken steps to address this by coarsening the data by pooling nodes. This transforms the node classification task into a graph classification task. For example as part of a phishing detection model, Zhang et al. (2021) sub-sample Ethereum transaction graphs to pass to an unsupervised hierarchical pooling technique called multi-channel graph classification. The technique is hierarchical because it concatenates the embeddings at each stage of pooling to form the final embedding. In another phishing detection task, Chen et al. (2020b) track accounts on Ethereum over time and then combine a gated recurrent unit (which regulates how much information the model 'remembers' from previous timesteps) with graph pooling to generate embeddings for graph classification. The idea is to detect how the behaviour of phishing accounts might change over time. Other methods for graph pooling include Diffpool (Ying et al., 2018), mincut pooling (Bianchi et al., 2019) and self-attention graph (SAG) pooling (Lee et al., 2019).

These pooling techniques are useful for scalability because they reduce the amount of observations to classify. However, they do not fully solve the scalability problem. For instance, if the pooling matrices are learned from the data (as they are in these algorithms) they can significantly increase the parameters the neural network has to learn and thus the complexity of the model.

A common challenge with the data used in these papers (which also extends to all financial transaction data) is dealing with class imbalance. This is a two-sided issue: the full transaction data is enormous at the same time as transactions labelled as high risk are only a small minority of all activity. This can make the training of large parameter models computationally infeasible and prone to overfitting, while also making it difficult to detect the minority class.

Sampling can help to address problems with class imbalance. One approach is to undersample the graph, for instance by sampling sub-graphs centred around central nodes of interest. Lin et al. (2020b), Zhang et al. (2021) and Xu (2020) use a technique called 'k-order subsampling' to sample observations that are k-hops removed from each central node. Wang et al. (2021) specify a similar method to construct Transaction Sub-graph Networks (TSGN), which Chen et al. (2020b) augment by taking into account the sequencing of those transactions. The minority class can also be oversampled. One technique to do this is to generate synthetic observations via a Generative Adversarial Network (GAN), which Agarwal (2020) use on a multi-class dataset of illicit Ethereum accounts. One benefit of using a GAN is that the synthetic data can robustify models against adversarial attack (which in this case could represent criminals adapting their behaviour to avoid detection) and the authors found that this method improves the performance of their model.

6 Data

Our data comes from three sources: the Elliptic Bitcoin dataset, a set of labelled entities and their (known) related Bitcoin addresses and transaction data obtained directly from the Bitcoin blockchain. We combine elements from each of these data sources to construct a transaction-level and address-level dataset. Details regarding the data sources and code are detailed in the README.Rmd file submitted with this thesis.

6.1 Elliptic Bitcoin Dataset

The Elliptic Bitcoin dataset consists of around 200,000 anonymised Bitcoin transactions analysed and released by Elliptic Co. to aid in the study of fraud detection on the Bitcoin blockchain (Weber et al., 2019).

The transactions are contained in one of 49 timesteps, which are derived from the actual timestamp of the transaction. Each timestep is spaced approximately two weeks apart and contains a single fully connected component of three hours' worth of transactions. The timesteps cover from the beginning of 2016 until October 2017. No transactions are connected between timesteps. The construction of the dataset in this manner means that it is stylised relative to the actual data generated from the blockchain (Figure 5).

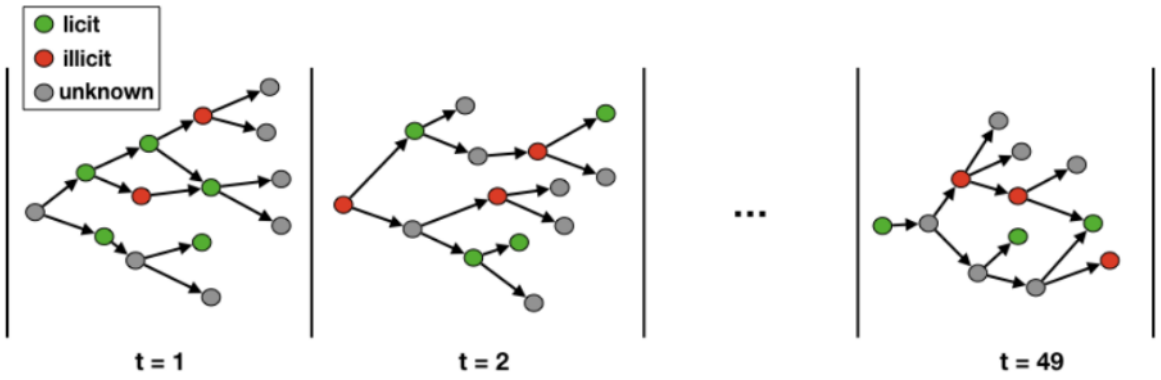

Figure 5: The Elliptic Data Set contains 49 connected components, from the earliest one denoted by the time step = 1 to the later one with a time step t = 49. Edges are directed from input to output.

Elliptic labelled transactions as illicit (2 per cent by volume), licit (21 per cent) and unknown (rest) based on the identity of the entity (or entities) that formed the inputs into the transaction. For each transaction, Elliptic calculated a set of 166 (proprietary) features based on the sample of transactions in the data set. A little over half of these features represent local information about the transaction — timestep, number of inputs and outputs into the transaction, fees, value — as well as global features describing the overall behaviour of the input and output addresses forming the transaction. The rest of the features represent neighbourhood information centred on the transaction of interest. These incorporate information from transactions one hop forward and backward from the transaction — maximum and minimum transaction amount, standard deviation and correlation coefficients between the local features mentioned above.

A notable exclusion from the Elliptic dataset is the individual input and output addresses involved in each transaction, which are needed to identify the offending entity (or entities) in a transaction. We discuss this further below.

6.2 Labelled entities (via Clovr Labs)

Clovr Labs have provided us with a set of 1600 known entities transacting on the Bitcoin blockchain. Entities are named and organised into categories.

These entities are labelled as high risk (35 per cent), low risk (5 per cent) and unknown (rest). 'Risk' in this context refers to the risk that an entity is involved in illegal activity. These labels broadly reflect our earlier description of high risk activities and are closer to how a bank may weigh up the risks of transacting with a counterparty that potentially has illegal dealings.

Each entity may use many addresses to transact on the blockchain. There are around 14m addresses linked to entities but only around 20k of these, corresponding to 60 entities, appear in the transactions in the Elliptic data set. Of the total number of addresses present in the Elliptic dataset, this represents only 2.7 per cent of input addresses and 2.5 per cent of output addresses (Figure 6).

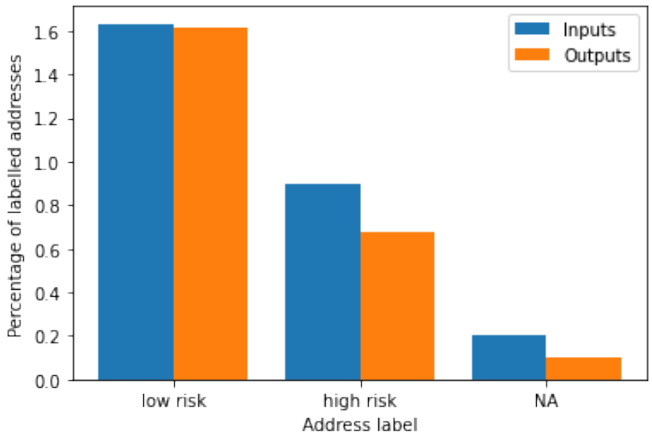

Figure 6: Percentage of Elliptic addresses that have Clovr labels

6.3 Bitcoin blockchain data

The Elliptic transaction data are anonymised — i.e. we do not know the underlying transaction features such as transaction hash, amount, the addresses involved, etc. — but Benzik (2019) has deanonymised (identified the hashes of the transactions) 99.5 per cent of the transactions by propagating identifiable traits from some transactions through the dataset. Using these deanonymised transaction hashes, we extract almost all of the transactions underlying the Elliptic dataset directly from the blockchain. Combining these data with the labelled entities data and the original Elliptic data set, we can then construct an augmented version of the (transaction-based) Elliptic dataset, as well as a novel (address-level) dataset based on the Elliptic transactions and labelled entities.

7 (Our) dataset

We combine elements of these three datasets to form two novel datasets: one at the transaction-level and one at the address level.

7.1 Labelling

First, we combine the labelling schemes from Elliptic and Clovr Labs to reduce the number of unlabelled observations in the data set. The Elliptic set is labelled at the transaction-level and the Clovr Labs entities are labelled at the address-level, so prior to combining them we have to translate each labelling scheme from the transaction- (address-) level to the address- (transaction-) level. This requires some simplifying assumptions.

For each transaction in the Elliptic data set, we can match the input and output address with those appearing in the Clovr labs entities data. Then taking a conservative approach to labelling (as perhaps a financial institution would), if any of the input addresses involved in the transaction are considered high risk then the whole transaction is labelled as high risk. We follow the Elliptic approach and focus on input addresses, since in AML the focus is primarily on identifying whether an counterparty you are going to accept money from is high risk, rather than the other way around. However, there is also a case for detecting high risk addresses from the output side (say to protect customers from scams) and our data could equally be constructed on that basis.

Then, we compare the label from both classification schemes. If either classifies the transaction as high risk, we label the transaction as high risk even if the labels disagree. For all remaining transactions, we apply a low risk label if one is available. This process resulted in 3.5 per cent of transactions classified as high risk, a 50 per cent increase from the original Elliptic labels. The share of low risk transactions

stayed roughly similar at 20 per cent (Figure 7).

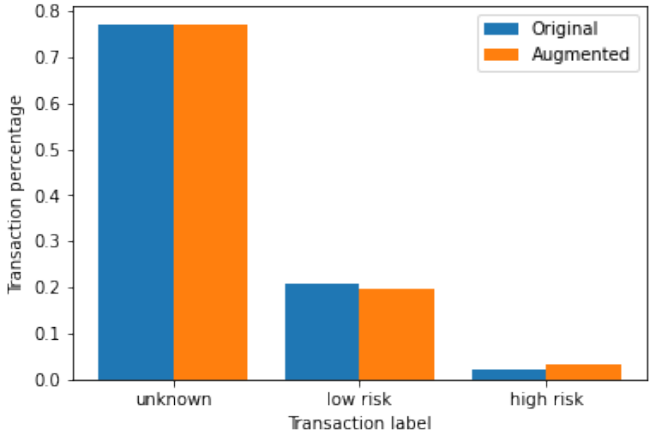

Figure 7: The share of the number of transactions in the Elliptic dataset split by original label and augmented label (after applying Clovr Labs' labelled data and also classifying any transaction as high risk if any of its input addresses are high risk)

For the address-level labels, we map the Elliptic transactions to their addresses and apply the transaction label to every input address. The Elliptic dataset classifies a transaction as illicit (licit) if the entity generating it is classified as illicit (licit). We take this to mean that all input addresses involved in a transaction carry the label of the transaction. Then, we apply the same logic as above and reclassify any transactions as high risk that involve at least one high risk classification between the schemes. This results in around 2 per cent of addresses with the high risk label and 20 per cent with low risk label. The rest of the addresses were left unlabelled. Figure 8 shows the resulting number of illicit/high risk addresses present per timestep in the Bitcoin blockchain dataset, split by whether their labels originated from the Ellitpic labels or the Clovr Labs labels.

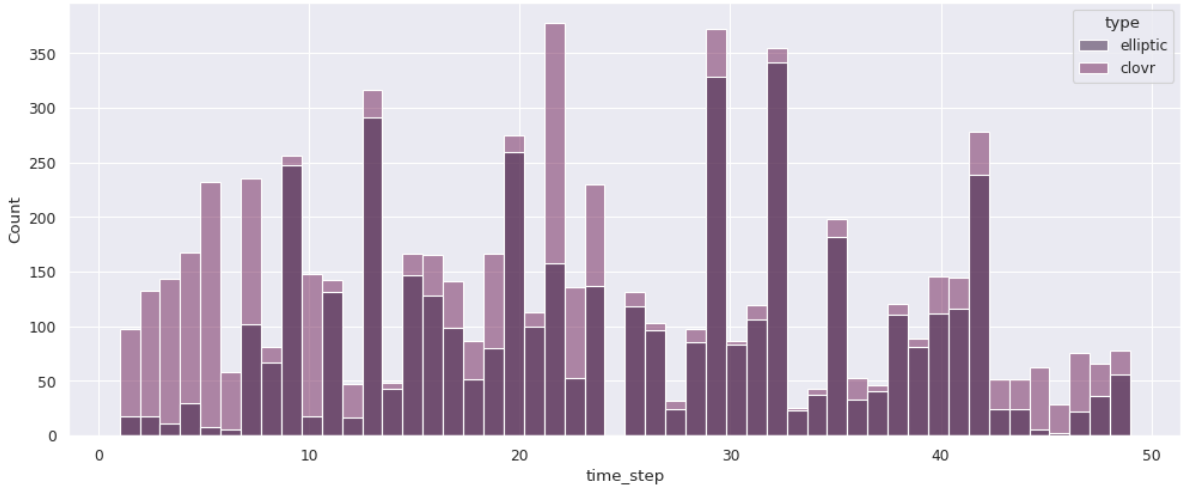

Figure 8: Number of high risk addresses per time step split by original Elliptic labels and augmented Clovr labels

7.2 Transaction-level dataset

The transaction-level dataset is simply the Elliptic dataset with its original features, but combined with the augmented transaction labels from Clovr Labs.

One downside of constructing the data set at the transaction level is that we must summarise much of the (potentially useful) input and output data underlying each transaction. This is partly offset by a

comprehensive approach to feature generation, but there is potential that we lose some of the transaction patterns that identify particular transactions as high risk.

When the transaction data is modelled as a graph, each node represents a transaction (with the incoming and outgoing edges representing the outputs from the preceding transaction and inputs to the following transaction respectively). Figure 9 (left) illustrates a simplistic example of two transactions that occur with only one input and one output address for each transaction. However, since there can be potentially multiple input and output addresses, if the model predicts an unlabelled node as high risk in this framework, there is no way to know which address (or addresses) participating in the transaction are responsible for this classification because there could be potentially hundreds of addresses involved in a single transaction.

In response to these shortcomings, we propose another way to organise the data based on the addresses in each transaction.

7.3 Address-level dataset

The address-level dataset is a more granular approach to summarising the Bitcoin data. Here we switch the focus from transactions to addresses. As far as we are aware, the literature has not used Bitcoin data at the address-level to train risk detection models. The dataset is a list of pairwise interactions between addresses in the transactions of the data set. The idea is that patterns between addresses (repeat interactions, amounts transacted, etc.) can help to identify if a given address is high risk. Now, when modelling the data as a graph, each node represents an address, and the incoming and outgoing edges represent the incoming and outgoing transactions. Figure 9 (right) shows the same simple two transaction example, but now modelled at an address-level.

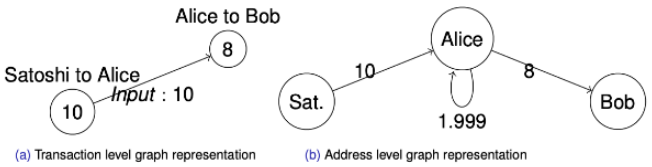

Figure 9: Satoshi sends Alice 10 Bitcoins and Alice sends Bob 8 Bitcoins

To arrive at the pairwise list of address interactions requires some manipulation of the raw Bitcoin transaction data. As discussed above, Bitcoin transactions can involve multiple inputs and outputs, so to calculate the pairwise interactions between addresses we need to look at every combination between input and output addresses in transaction. For example, if a transaction has three input addresses and two output addresses, it will have six pairwise address combinations. Next, we allocate the input and output values and fees from the transaction to each pairwise combination. We do not know how much value from each input address was allocated to each output address, so we allocate the inputs (net of the fee) proportionately to each address's share of the total output (Equation 1). Fees are allocated in the same way (Equation 2). We remove any addresses that do not transact with another address, to preserve the connectivity of the network.

$$value_{ij} = (\sum_{i=1}^{n} input_i - total fee) * \frac{output_j}{\sum_{j=1}^{m} output_j} \tag{1}$$

$$fee_{ij} = total fee * \frac{output_j}{\sum_{j=1}^{m} output_j} \tag{2}$$

for each input-output pair ij, where $i = 1, ..., n$ are the inputs into a transaction and $j = 1, ..., m$ are the outputs.

In addition to these pairwise features, we also calculate a set of address-level features. These include variables such as: total, minimum, maximum, mean and medium amounts transacted by an address; total fees and the fee share of the total inputs generated by an address; the number of timesteps an address appears in. These features are calculated separately for transactions where an address was involved as an input and an output. We also calculate the number of repeat transaction partners for each address and the mean, median and maximum number of interactions with another address. We do not create any information that indicates which specific timesteps the address transacted in, since if a financial institution was looking to classify transactions in real time, there would be no prior information on the number of high risk addresses transacting at that given point. Overall we create 24 features related to each address and each is normalised to have zero mean and unit variance.

Looking at the pairwise interactions between addresses is computationally expensive relative to looking at transactions, since each transaction expands from a single row of data to many rows. Since our address-level dataset is basically an edge list of a directed graph, we can use k-order subsampling to sample address-level subgraphs (Lin et al., 2020a), meaning that neighbours that are 'k' or less nodes in distance from each source node will be included (Figure 10). Our edge list is directed, such that the subgraphs only contain edges between two nodes in the input to output direction.

Our subgraphs use sampling of order 3, given that previous studies have shown that taking more than three or four convolutions out from a central node can produce suboptimal results as a result of over-smoothing (Chen et al. (2020a)). We discuss over-smoothing further in the modelling section. We also aim to lessen the impact of the class imbalance in the dataset by undersampling the low risk subgraphs in the training set.

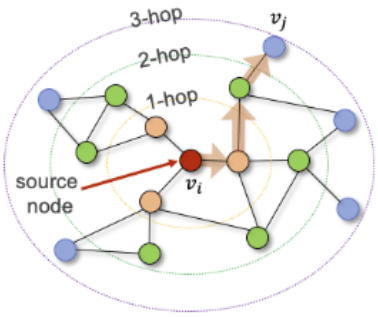

Figure 10: K-order sub-sampling. Source: Xu (2020)

Each subgraph is centred around a particular address, enabling us to classify at the (sub)graph level using our address-level label. This approach allows us to leverage the local network structure around an address and combine it with the features of the address of interest and its neighbours. For node features, we use the address-level features calculated above while for edge-level features we include the (proportionate) transaction and fee amount, an indicator for whether the relationship between addresses is two way and the number of interactions between those addresses. To avoid using a more complex multi-edge framework and to help with computational tractability, we sum multiple interactions between addresses into one edge and represent these multiple edges through an edge attribute. Edges of the sub-graphs are directed in the input direction. The edge features are also standardised to have zero mean and unit variance.

8 Model Methodology

In this section, we outline the classifiers used on each dataset. We include some baseline models as well as some more complex GNN architectures that take graph data as their data. We have chosen two

baseline models: a basic linear classifier (logistic regression) and a non-linear classifier (random forest). These models are an appropriate baseline because they are efficient to train and both carry an element of explainability: the coefficients from the logistic regression and the feature importances for the random forest[5]. However, they also treat observations as identically independently distributed (iid) which is a questionable assumption when our data are structured as a graph. The baseline models will serve as a benchmark for our graph-based models to beat.

For the baseline models, the label of each transaction is the dependent variable while the node features are passed in as the set of independent variables. For each baseline model we use the default hyperparameter seeting.

We also try several graph-based neural network models, including a vanilla graph convolutional network (GCN) (Kipf and Welling, 2017), a graph attention network (GAT) (Veličković et al., 2017), (Brody et al., 2021) and a model called APPNP which leverages the relationship between GCNs and personalised page rank algorithms (Gasteiger et al., 2018). Since these models are perhaps less well-known than our baseline models, we provide a brief description of each below.

8.1 Graph Convolutional Network (GCN)

A GCN is an autoencoder method applied to graph data. It applies convolutions over each node in a graph that respect the ordering of the nodes (permutation invariance) and also the graph's non-euclidean structure. A GCN consists of two steps: update and aggregate (Equation 3). In the aggregate step (inside the parentheses), the GCN convolves over itself and each of its neighbouring nodes, collecting information from its own feature vector and the feature vector of each of its neighbours ($\mathbf{h_j^{(k)}}$). This framework also allows the user to concatenate edge attributes with node features. The information aggregated from neighbouring nodes is usually normalised in some way, say by the degree of the node (in Equation 3 we use the normalisation proposed by Kipf and Welling (2017)). The normalisation helps to avoid high degree nodes dominating the embeddings of all the nodes in the graph. Learnable weights ($\mathbf{W}$), shared over all nodes, 'learn' how to aggregate the information from each of the node (plus edge) feature vectors into new embeddings for each node. This step is called 'message passing' because a node is receiving 'messages' from itself and each of its neighbours. Then in the update step (σ), the new embeddings for each node are passed through a non-linearity, such as a Leaky ReLU and updated to a new value.

The aggregated embeddings are then passed to a classifier (such as a fully connected linear layer) and the weights of the GCN layer are updated via backpropagation. In an architecture, several GCN layers can be stacked one after the other and this is analogous to convolving a certain number of 'hops' out from a central node. However, it is recommended not to exceed 2-3 convolutions to avoid over-smoothing (Chen et al., 2020a). Over-smoothing occurs when a node receives messages from so many nodes in the graph that its embedding loses its local structure. Since this occurs for every node, the embeddings tend toward the same value for all nodes.

GCNs do not explicitly incorporate the direction of the edges in the message passing process, though direction is still important in our data since it determines the nature of the edge features (which measure input address to output address flows). Direction could be incorporated into the model using the method outlined by Dernbach (2018).

$$h_i^{(k)} = \sigma\left(\sum_{j \in N(i), i} \mathbf{W} \frac{\mathbf{h}_j^{(k-1)}}{\sqrt{deg(i) + deg(j)}}\right) \tag{3}$$

[5]In a random forest, the importance of a feature is the share of trees where it is used as the first split in a decision tree. This is partly stochastic, since for each tree random forest samples a subset of features to build the tree from. If a feature provides the first split in the decision tree, it provides the largest decrease in impurity between the classes, making it the most important feature for classification.

We base our implementation of the GCN on code from Li (2014).

8.2 Graph Attention Network (GAT)

The GAT model adds an attention mechanism to the basic GCN layer (Equation 4). The attention mechanism is a vector of learnable weights (α_{ij}) that determine how much 'attention' to pay to the embedding of each node in the aggregate step. Multiple attention heads are available to learn different types of relationships between nodes (with the coefficients over multiple heads concatenated or averaged). Where the GCN layer is analogous to taking a uniform average of the embeddings in aggregate step, the GAT layer transforms this to a weighted average where the weights are the attention coefficients. In practice, attention is useful because it allows the model to focus more on the node's own embeddings, rather than treating it as equally important as neighbouring nodes.

$$h_i^{(k)} = \sigma\left(\sum_{j \in N(i),i} \mathbf{W}\alpha_{ij} \frac{\mathbf{h}_j^{(k-1)}}{\sqrt{deg(i) + deg(j)}}\right) \tag{4}$$

In our implementation, we use the updated graph attention mechanism discussed in (Brody et al., 2021). This conditions the attention weights on the central node, enhancing their interpretability (e.g. for a given graph, we can see for each node's embedding which other nodes it is paying most attention to).

8.3 APPNP

This model first makes predictions from a node's own features, which are then propagated through the graph using Personalised Page Rank: a method that determines the importance of neighbouring nodes using an algorithm similar to a random walk. However, similar to the concept of a gated recurrent unit the model incorporates a 'restart' probability (α) that the walk jumps back to the central node (Equation 5).

$$h_i^{(k)} = (1-\alpha) \sum_{j \in N(i),i} \frac{h_j^{(k-1)}}{\sqrt{deg(i) + deg(j)}} + \alpha f_\theta(h^{(0)}) \tag{5}$$

where f_θ indicates a single layer neural network with parameter set θ, and α indicates the likelihood of the random walk jumping back to the central node.

Our code for this model is adapted from Arcos-Diaz (2019).

8.4 Utility layers

In addition to these modelling frameworks, we use a number of utility layers to help with stability and performance. We note them briefly below.

- **Graph normalisation**. Graph normalisation layers are used at each step of our architectures to help speed-up the training of the model. Graph normalisation is a graph-specific version of batch normalisation and is shown by Cai et al. (2021) to outperform batch normalisation methods with graph data. Some form of normalisation is standard across neural network architectures.
- **Pooling**. For the address-level models our task is graph classification, so we need a way to summarise the embeddings learned for each node to a single graph embedding. A global pooling layer at the end

of the network helps to facilitate this. We use global mean pooling, which takes an average of the embeddings in the graph. However, we also try some intermediate pooling layers, which attempt to learn how to pool embeddings through the network. Specifically, we use self-attention graph (SAG) pooling which learns attention weights to determine the contribution of each embedding to a cluster Lee et al. (2019). We also use mincut pooling, which applies a form of spectral clustering over the nodes Bianchi et al. (2019). The intermediate pooling layers help to reduce the information loss from taking a global mean across all nodes in the subgraph. However, they also add parameters for the neural network to learn.

- **Dropout**. We use dropout to help avoid overfitting.

8.5 Train-test split and sampling

We split our data using a 90/10 train/test split. Due to the class imbalance discussed earlier, we perform undersampling for all models to balance the training data toward high risk observations. Our test data is left unbalanced.

8.6 Transaction-level models

Tables 2 and 3 shows each of the models trained using the transaction-level data as well as their hyperparameters. For the graph-based models, we provide sketches of the architecture in Figure 11 and Appendix A.

Model	Parameter restrictions
Logistic regression	L2 penalty
Random forest	Sklearn default

Table 2: Baseline models hyperparameters

Model	Hidden dims	Attention heads	Layers	Weight decay	Learning rate	Class weights	Dropout rate	Teleportation probability	Power iterations	Activ function
GCN	16	NA	2	0.0005	0.01	0.66	0.5	NA	NA	Leaky Relu
GAT	8	8	2	0.0005	0.005	0.8	0.6	NA	NA	Leaky Relu
APPNP	256	NA	2	0	0.001	0.7	0.2	0.2	20	Leaky Relu

Table 3: Graph based models hyperparameters - transaction data

The APPNP network architecture is a common multi-layer perceptron that uses the PageRank algorithm implemented in the APPNP convolution to perform message propagation. We use two layers, which both include a leaky ReLU activation and dropout layer.

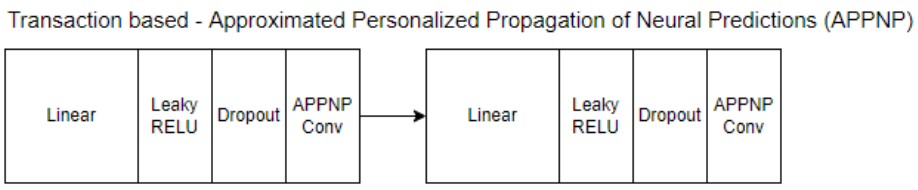

Figure 11: APPNP network architecture

8.7 Address-level models

Tables 2 and 4 show each of the models trained using the address-level data as well as their hyperparameters. For the graph-based models we provide a sketch of the architecture (Figure 12) and Appendix A.

Model	Hidden dims	Attention heads	Layers	Weight decay	Learning rate	Class weights	Dropout rate	Int pooling	Global pooling	Activ function
GAT	256	10	2	0.0005	0.001	None	0.5	SAG	Mean	Leaky Relu
GATMinCut	128	10	2	0.0005	0.005	None	None	NA	MinCut	Leaky Relu

Table 4: Graph based models hyperparameters

Our graph attention configuration combines two graph attention layers with leaky ReLU activation, followed by graph normalization. On the second layer we include a dropout stage and an intermediate SAG pooling layer. Finally, we perform global mean pooling to aggregate the embeddings up to the graph level and pass this to a fully connected linear layer for classification.

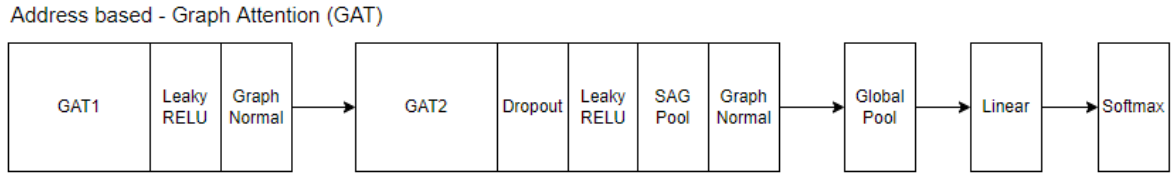

Figure 12: Graph attention network (GAT) architecture

9 Results

9.1 Transaction-level model

First, we discuss results for the transaction-level data with the augmented labels. Across all models we observe a decline in performance, including recall, relative to the original Elliptic dataset (Tables 5 and 6). Accuracy in the original dataset is very high relative to our augmented version, but this is not a reliable metric for performance given the class imbalance present in the test data.

In the original Elliptic analysis, the authors also run their models separately on the data contained in each timestep. Toward later timesteps they observe a large decline in performance, which they attribute to a decline in the number of high risk labels because of the closure of a large Darknet site (Weber et al., 2019). To see if our augmented labelling scheme solves this issue, we replicate this analysis. Figure 13 shows that the augmented dataset exhibits the same deterioration in performance in later timesteps, although we note from Figure 8 that the original results were sometimes based on very few high risk observations in each timestep.

This means that the extra labels from Clovr Labs were unable to offset the closure of the Darknet market. This is perhaps surprising, given that the addition of the Clovr Labels increases the size of the high risk class by 50 per cent (from 2.2 per cent to 3.3 per cent of all transactions).

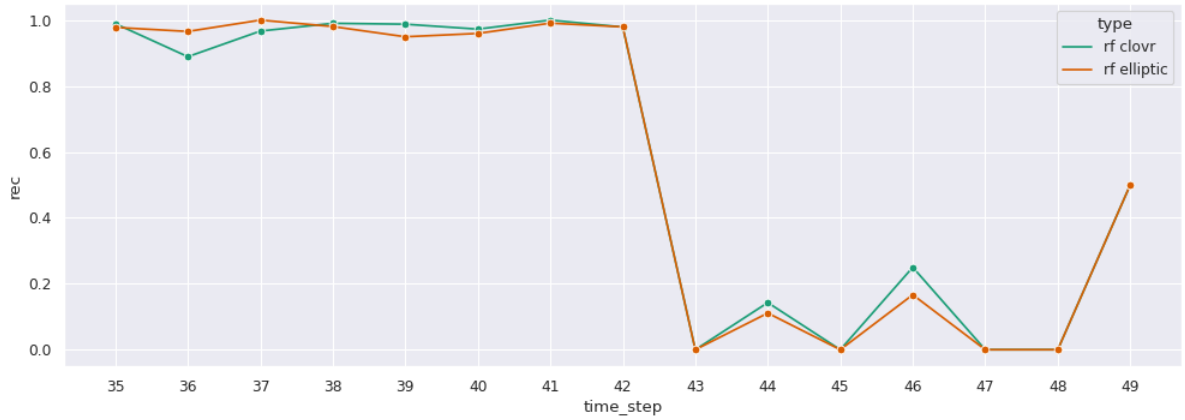

Figure 13: Random forest recall for clovr and elliptic labels by timestep

Next, we consider the performance of the baseline models compared to the models that account for graph structure. Tables 5 and 6 show the performance of the best baseline and graph-based model for our dataset versus the original Elliptic dataset, while Table 7 shows the results for all models using our dataset. The GNN models do not show any improvement in performance, including the recall, relative to the best baseline model (random forest). For the APPNP model we observe some modest improvements in performance, but the results are still not sufficient. Given that we are unable to improve the performance of the transaction-level model on our key metric, we assert that this dataset is not well-suited to our classification task and would not meet the requirements of a risk detection model for a financial institution. This is unfortunate because the transaction-level models are efficient to train and scalable to large amounts of data.

model	rec	pre	f1	acc
rf	0.53	0.96	0.68	0.96
rf elliptic	0.72	0.95	0.82	0.98

Table 5: Transaction-level test scores for random forest models

model	rec	pre	f1	acc
APPNP	0.54	0.69	0.61	0.94
APPNP elliptic	0.71	0.67	0.69	0.96

Table 6: Transaction-level test scores for APPNP models

model	rec	pre	f1	acc
logit	0.68	0.15	0.25	0.63
rf	0.53	0.96	0.68	0.96
GAT	0.41	0.57	0.48	0.92
GCN	0.45	0.32	0.38	0.86
APPNP	0.54	0.69	0.61	0.94

Table 7: Transaction-level test scores for all models

9.2 Address-level models

For the address-level data we run two versions of the baseline models: one classifies nodes using the node feature matrix as input and the other classifies edges that have a high risk input using the edge attributes as input. Then the GNN models take in both the node features and edge attributes in the construction of the sub-graphs.

The baseline models using the node features outperform those using the edge attributes (Table 8). Although the highest recall score is given by the logistic regression model on the edge features, the result is essentially meaningless — the model is classifying almost all observations as high risk (evidenced by the extremely low precision). Therefore, we disregard the results of the edge-based models and posit that the baseline to beat is the random forest trained on the node features, with a recall of 79 per cent and a precision of 28 per cent.

For the GNN models, the GAT architecture and GAT augmented with mincut pooling exhibit similar results (Table 8). However, the addition of mincut pooling renders the model highly unstable and more computationally intensive to train. In some cases, introducing new data to the mincut algorithm requires

re-tuning of the hyperparameter for the algorithm to converge. As a result, we choose the GAT architecture as our preferred GNN.

The GAT model shows an improvement in recall of around 4 percentage points (79 to 83 per cent) over the baseline random forest, with a slight improvement in precision. The GAT with the mincut pooling layer performed similarly, but required significantly longer to train. Therefore, in terms of performance the GAT model trained on the address-level data is our best candidate for a financial institution.

As noted earlier, we corrected for class imbalance by undersampling subgraphs in our training set, but not our test set. To understand how much this class imbalance is driving results, we also passed a balanced test set to the models. When the test data is balanced we see a significant improvement in precision and accuracy and also a slight improvement in recall (Table 10). This is an intriguing result and suggests that if we had some way to balance unlabelled data (say by discarding a large number of addresses that are very likely — or even known — to be low risk) we could substantially improve the utility of our candidate model. We discuss possibilities for this further in the analysis section.

model	rec	pre	f1	acc
logit nodes	0.44	0.17	0.25	0.73
logit edges	0.96	0.08	0.15	0.20
rf nodes	0.79	0.28	0.41	0.77
rf edges	0.77	0.12	0.21	0.56
GAT	0.83	0.26	0.40	0.74
GAT + MinCut	0.83	0.26	0.39	0.74

Table 8: Address level scores for all models

10 Analysis and Recommendations for Future Work

In this section we analyse our results with respect to the three key requirements of financial institutions: performance, scalability and explainability, and also provide recommendations for future areas of research and improvement.

10.1 Performance

So far we have discussed the performance of our models with respect to the counts of high risk addresses the model is able to correctly classify. However, the loss function of a financial institution is more sophisticated than this in the sense that they care also about the value of the high risk transactions they are able to capture. A risk detection model would be ineffective if it failed to capture the largest value transactions. We recreate our recall and precision measures for our candidate model based on the value of high risk transactions. By value, this results in a recall of 96 per cent and a precision of 74 per cent, substantial improvements from our original metrics (Table 9). We also see that our baseline random forest model does much worse when we measure recall and precision by value. If we compare these results to the best recall we found in the literature (albeit based on a transaction-level dataset), our results also outperform Arcos-Diaz (2019). These results are encouraging and suggest that, at least with our candidate model, that the false negative and false positive transactions are at the lower end of the value distribution and probably of little consequence to the financial institution.

	Recall	Recall(by value)	Precision	Precision (by value)
GAT (candidate)	0.83	0.96	0.26	0.74
RF (baseline)	0.79	0.59	0.27	0.58
Arcos-Diaz(lit)	0.72	-	0.92	-

Table 9: Summary of key results (test data)

10.2 Scalability and performance - a 'pipeline' for risk detection

A criticism of risk detection models for cryptocurrencies is that they are trained on highly stylised datasets, such that they are not scalable to 'real' volumes of crypto data. Our version of the Elliptic dataset is no exception: it consists of 200,000 transactions, amounting in total to just 6 days worth of transactions (in 2016-17 when crypto was much less traded than it is now).

To address this criticism, our goal was to propose a two-step 'pipeline' methodology that financial institutions could employ to scale their risk detection models. The first step of this approach would act as a filter and have the capability to process large amounts of data. It would discard many transactions and/or addresses that are highly likely to be low risk and keep observations where this distinction was not so clear. This is analogous to a model with a very high recall, but low precision. Then, as a second step, the observations that passed through the filter would be subject to a more sophisticated model to decide whether they are high or low risk. Since the first step of the pipeline would discard an ideally large share of low risk transactions, the model in the second step would not have to deal with such a large class imbalance problem.

We aimed to achieve the first step of this pipeline using the transaction-level data, which is very fast to train a model on. However, we were unable to achieve sufficient performance in the recall of models based on transaction data. For the second-step of the pipeline, our GAT model trained on the address-level data is a suitable candidate for two reasons. First, our subgraphs contain a lot of information about an address and its relationship with neighbouring addresses, giving the user greater opportunities to explain why a given address is flagged by the model as high risk. Second, although in the main results we presented the precision of the model was low, we note that when we test the model on a balanced dataset its performance improved to around 85 per cent for precision and recall.

model	Recall	Precision	Accuracy
GAT	0.85	0.86	0.85
GAT + MinCut	0.84	0.84	0.84

Table 10: Address level test scores with balanced dataset

Therefore, the key to building a successful pipeline is to find an approach that can efficiently filter for many low risk addresses. One suggestion is for financial institutions to borrow from techniques they use in AML compliance in traditional financial markets. For instance, financial institutions that perform checks on their counterparties often share this information using 'whitelists' to avoid duplicating verification efforts. Sometimes third-party businesses may even perform these checks on behalf of financial institutions and so have a large database of low risk counterparties from many institutions. These whitelists can quickly scale if the network of participating institutions is large. For instance, financial institutions could share information with compliant cryptocurrency exchanges to build a whitelist. However, at the current time such a development seems challenging in a cross-border environment — countries may not trust the whitelist of countries with less sound institutions. Nevertheless, if a whitelisting filter could help balance data passed to our address-level model, then its performance may meet the requirements of a financial institution.

10.3 Scalability

The novel transformation of the Elliptic dataset from transactions to addresses produced much improved results but also introduced a lot of computational complexity into the modelling approach. For instance, translating the 200,000 transactions into address-level subgraphs resulted in an edge-list of 12 million rows. We handled this increase in the scale of the data by undersampling the low risk subgraphs used to train the model. However, managing the incoming data on an ongoing basis is clearly computationally challenging and it is unclear the extent to which the risk appetite of the financial institution would allow for undersampling of incoming data. Although our pipeline suggestion above is a practical solution in helping to address this issue, more computational power and storage would nevertheless be required to scale our address-level model for production.

10.4 Explainability

A key challenge for financial institutions is to be able to explain the reasons why the model makes the predictions that it does. Each counterparty that an institution declines to transact with presents an opportunity cost in the form of lost revenue and so the institution needs some evidence as to why the counterparty presents a risk. Since our candidate model is a highly parameterised, non-linear neural network it is not an ideal framework for explainability and so this is not a requirement we are able to address. In place of that, we present some analysis from our outputs that could help to provide reasoning for the classification decisions.

First, we use the feature importances calculated from our baseline random forest models. Feature importance measures the share of decision trees where a node feature was used in the first split (which indicates it was the best able to separate the classes). One warning with this approach is that the random forest model treats observations as iid, so its validity may not necessarily extend to the relationships driving results in our GAT model.

For the transaction-level data, no one feature drives a significant share of the importance (the maximum contribution highest is 3.5 per cent). However, 80 per cent of the importance accumulates to 40 per cent of the features. The exact nature of the features is unknown (since they were pre-calculated by Elliptic). However, features based on the local and neighbourhood information around a transaction contributed 78 per cent to the overall importance (as opposed to global information relating to the addresses making up the transaction). These importances run contrary to our address-level model, where aggregated data relating to an address (the node features) are useful and is perhaps a reason for why the models trained on each dataset leads to different outcomes.

For the address-level data, Table 11 shows the feature importances for the random forest trained on the node features. Of the 24 features, the first 9 account for around 80 per cent of the importance and mostly relate to fees, transaction counts and transaction amounts.

feature*	importance	cumulative importance
fees paid (input)	0.12	0.12
fee as share of amount transacted (input)	0.11	0.23
minimum amount transacted (input)	0.11	0.33
degree (input)	0.10	0.43
median amount transacted (input)	0.09	0.53
total amount transacted (input)	0.09	0.62
maximum amount transacted (input)	0.09	0.71
mean amount transacted (input)	0.08	0.79
fee as share of amount transacted (output)**	0.03	0.82
other	0.18	1.00

Table 11: Feature importance from random forest address level model

∗[6] ∗∗[7]

Features related to the fees paid by an input address account for around 20 per cent of the overall feature importance. Other important features include statistics related to the number of transactions an address was involved in, the number of other addresses it transacted with (degree) and the amounts transacted.

Table 12 below shows some summary statistics, split by high and low risk addresses, as well as between true positives and false negatives. We observe that high risk addresses tend to send lower and receive lower amounts (i.e. as an input and output address) than low risk addresses. Furthermore, they tend to have a lower degree (and lower number of nodes in their subgraph) although this does not affect the number of transactions they engage in. Other than the fact that high risk addresses are used less regularly and for smaller amounts, these summary statistics do not give much of an insight into what could be driving the behaviour of high and low risk addresses, and none of these differences are statistically significant.

Next, we discuss these statistics with respect to the true positives and false negatives from our candidate model. We see that addresses with certain characteristics are misclassified for both the high risk and low risk addresses. For example, on average the model misclassifies high and low risk addresses with a higher input degree and lower input fee share. From this we might infer that such addresses behave similarly to each other and hence the model has trouble distinguishing between them. The output degree of misclassified high risk addresses is on average lower than the global mean for both high risk and low risk addresses. This also indicates that the model is not able to interpret the behaviour of addresses that only receive payments from a low number of addresses. These results are significant at the 5 per cent level.

Finally, we see that both the average total transaction input and output of misclassified high risk addresses is lower than the correctly classified ones (although were not statistically significant). This is encouraging for a model of our purpose, as it indicates the missed high risk activity is of lower value than those the model correctly classified. These figures also explain our increased recall score when weighting by transaction value.

type	number of nodes	degree (input)	degree (output)	transactions (input)	transactions (output)	fee as share of amount transacted (input)	fee as share of amount transacted (output)
high risk	63	9.16	2.32	1.19	0.51	0.10	0.00
low risk	67	9.53	2.84	1.22	0.55	0.10	0.01
high risk matched	63	7.24	2.62	1.15	0.50	0.12	0.00
high risk mismatched	61	14.98	1.41	1.33	0.55	0.03	0.00
low risk matched	69	7.83	3.05	1.19	0.53	0.11	0.01
low risk mismatched	62	15.61	2.06	1.32	0.64	0.04	0.00

type	total amount transacted (m) (input)	total amount transacted (m) (output)	mean amount transacted (m) (input)	mean amount transacted (m) (output)
high risk	182	137	42	62
low risk	324	211	88	127
high risk matched	206	166	44	70
high risk mismatched	110	48	36	39
low risk matched	339	228	93	140,
low risk mismatched	271	151	70	82

Table 12: Summary statistics for test data, split by high risk, low risk and correctly and incorrectly predicted by the GAT model. Note that transaction amounts displayed are in Satoshis. There are 10^8 Satoshis in 1 Bitcoin.

Although we have made a few observations from our summary statistics, it is difficult to weave these into a coherent narrative and the insights they provide are rather limited since we are taking global averages over all addresses. We also acknowledge that the validity feature importances output from the random forest may not extend to the behaviour of our GAT model. Future work should aim to address this key explainability requirement in more detail.

[6] Input and output refer to the role of an address in a transaction. As an input, the address participated in generating the transaction while as an output the address was on the receiving end of a transaction. Our subgraphs are constructed with edges flowing between addresses in the input direction only.

[7] Fees in transactions where the address appeared as an output. Fees were paid by the input address.

One possibility, which we did not explore, is to extract insights from the attention coefficients learned in our GAT model. For each node, the GAT model stores a vector of coefficients that indicates how much it weighted its own embeddings and the embeddings of its one-hop neighbours. In assessing why a particular address may be high or low risk, it could be informative to see which connections between addresses the model found important in determining its prediction.

10.5 Explainability — Exchanges

The Clovr Labs dataset tags addresses with the name and category of the entity, where it is known. Figure 14 shows the count of addresses in the categories appearing in our dataset (regardless of their label). These make up around 12 per cent of our subgraphs, with the rest unlabelled. The majority of the labelled subgraphs relate to exchanges (around 80 per cent), while a smaller number fall into other categories.

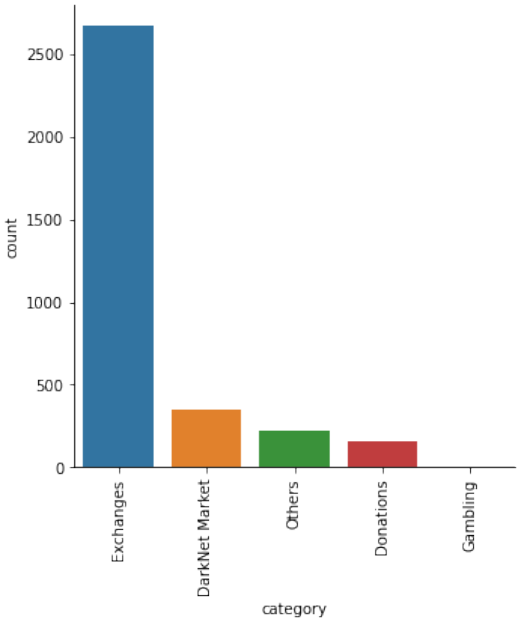

Figure 14: Count of Clovr Labs tagged addresses by category

Since exchanges dominate our labelled entities can explore their role further by looking at some example subgraphs. Figures 15, 16 and 17 randomly sample subgraphs of an exchange address from the test set that are low and high risk respectively (Appendix B includes example subgraphs of the three other most present categories). In the figures, the orange node is the central address in the subgraph, while the yellow node has the highest (input) degree. The high risk plots differentiate between exchanges that our GAT model classified correctly and incorrectly. Anecdotally, we can see there are roughly two types of exchange addresses: ones that interact with many other addresses and others that interact with few other addresses. Both types of subgraphs appear in all of our plots and there doesn't appear to be any obvious difference visually between high and low risk exchanges. However, we can see that the model tends to correctly predict high risk exchanges with many interactions and performs less well with those with relatively few.

The summary statistics in Table 13 also support these findings — the average input degree of correctly predicted exchanges is 16, whilst the average for incorrectly predicted exchanges is 10. Similarly, we noted in the previous section that the model was less successful at predicting addresses with lower input fee shares. Table 13 also shows that the incorrectly predicted exchanges display much lower average input fee shares. Both of these results are statistically significant at the 5 per cent level.

More generally, the model correctly predicts 88 per cent of low risk exchanges, compared to only 83 per cent for high risk exchanges. The gap here implies that the model has some difficulty identifying high

risk exchanges, relative to low risk ones. One hypothesis from these observations could be that high and low risk exchanges behave similarly and thus the model may struggle to differentiate between them. If this is the case, then it is a challenging problem for the model.

type	number of nodes	degree (input)	degree (output)	transactions (input)	transactions (output)	fee as share of amount transacted (input)	fee as share of amount transacted (output)
high risk	104	14.92	4.10	1.15	0.59	0.21	0.00
low risk	159	15.64	4.34	1.22	0.56	0.24	0.01
high risk matched	102	15.75	4.34	1.12	0.56	0.25	0.00
high risk mismatched	116	9.75	2.68	1.32	0.71	0.01	0.00
low risk matched	167	16.48	4.52	1.23	0.55	0.24	0.01
low risk mismatched	109	9.69	3.12	1.18	0.65	0.20	0.01

type	total amounttransacted (m) (input)	total amount transacted (m) (output)	mean amounttransacted (m) (input)	mean amount transacted (m) (output)
high risk	499	79	40	24
low risk	323	66	28	16
high risk matched	518	88	41	27
high risk mismatched	402	22	34	6
low risk matched	336	73	29	17
low risk mismatched	235	15	22	5

Table 13: Summary statistics for exchanges data, split by high risk, low risk and correctly and incorrectly predicted by the GAT model

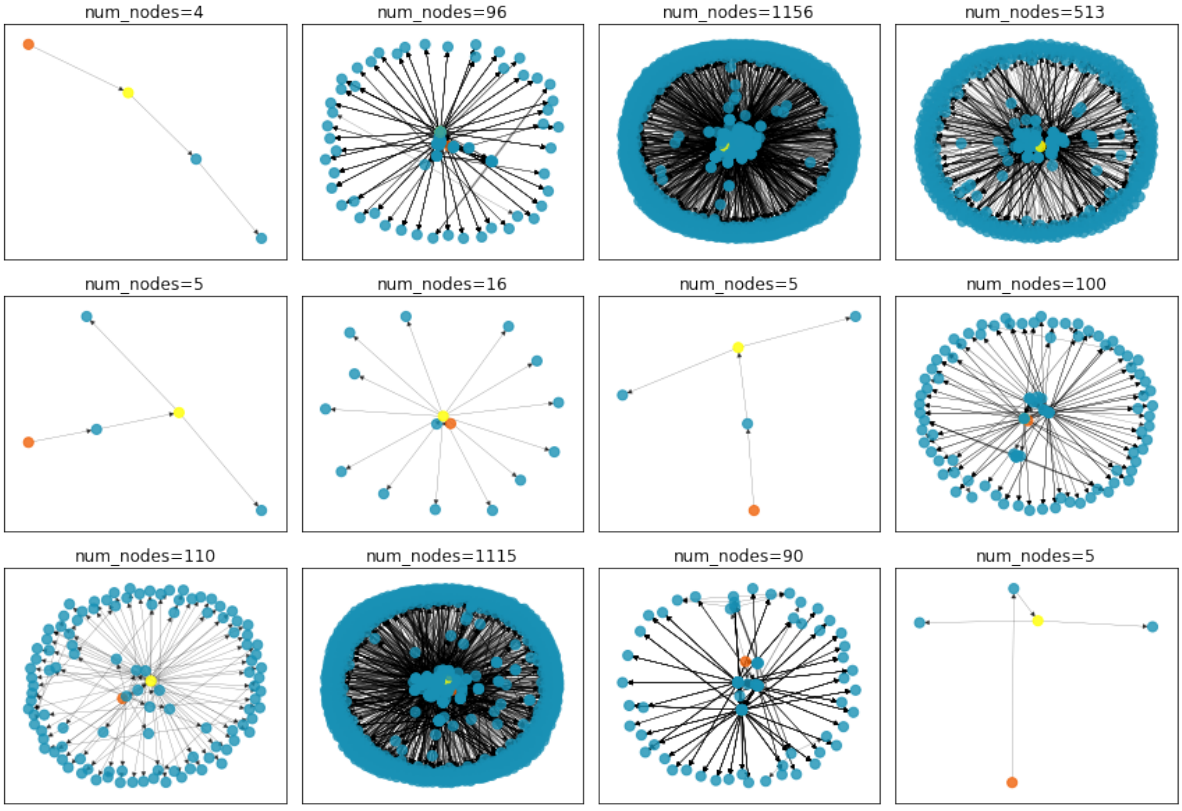

Figure 15: Subgraphs for low risk exchanges

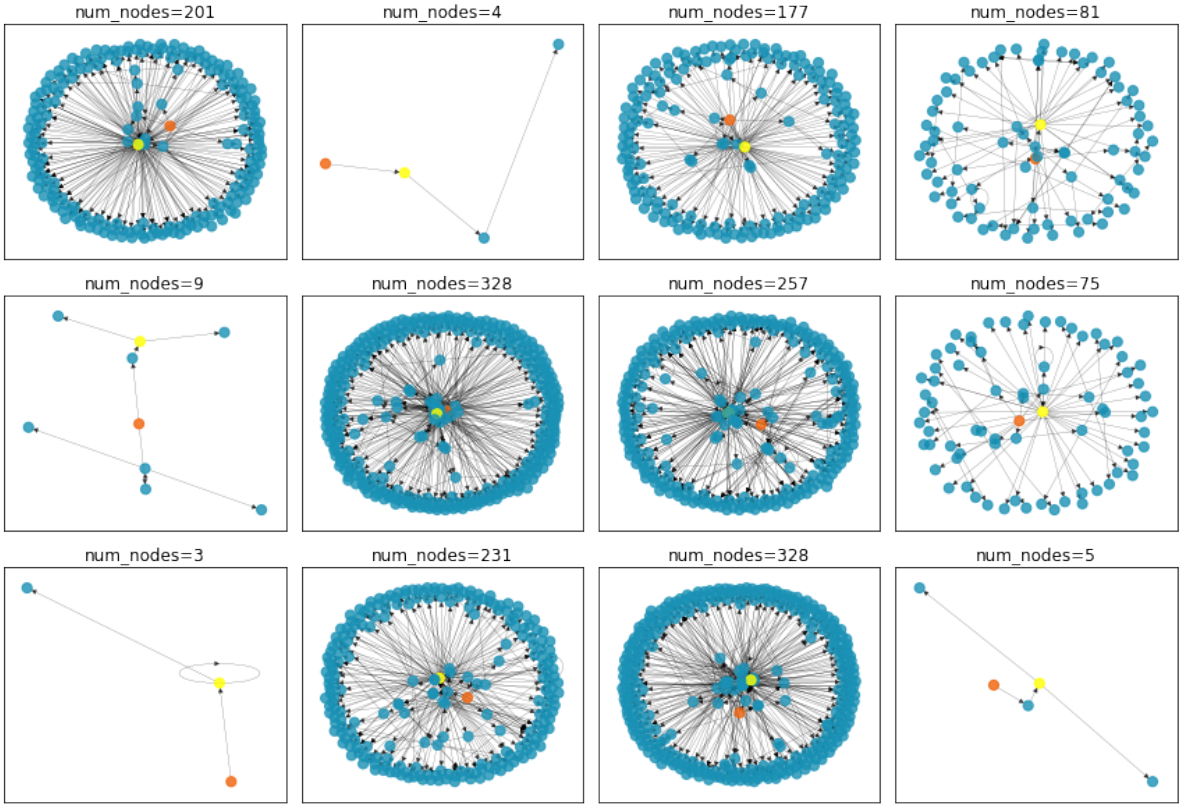

Figure 16: Subgraphs for high risk exchanges correctly predicted

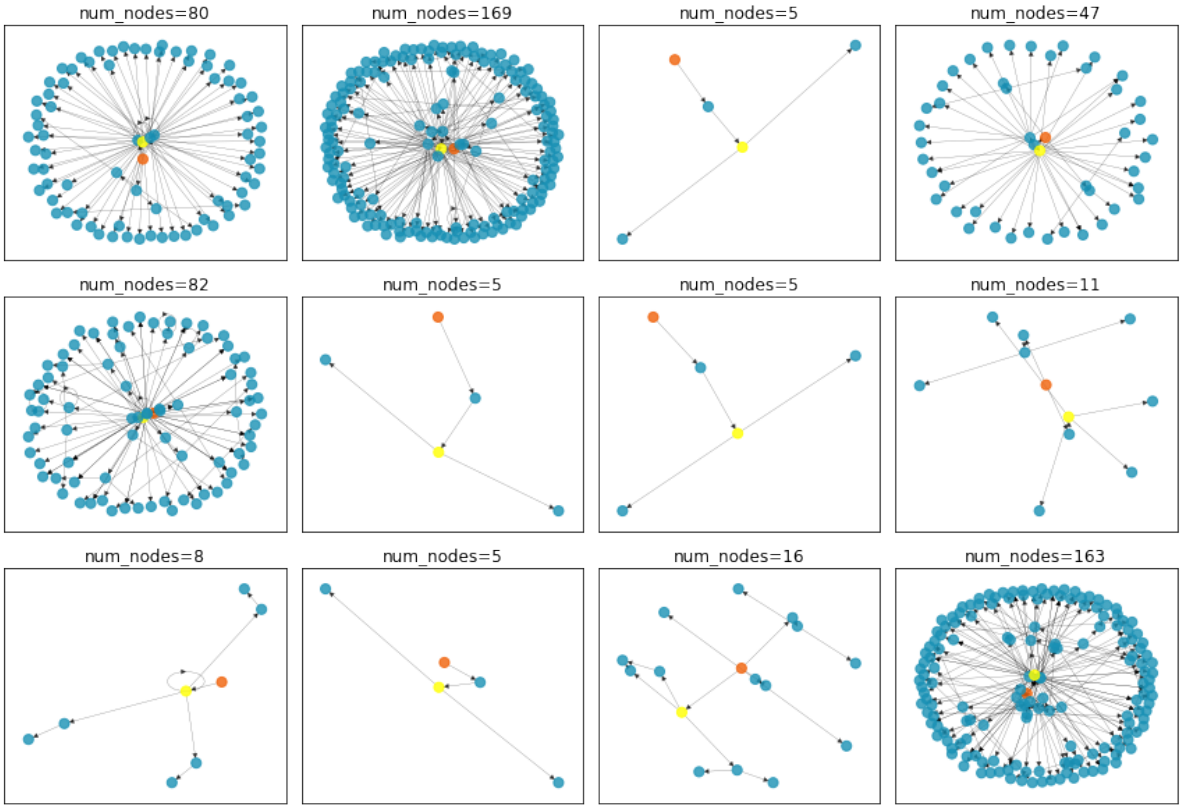

Figure 17: Subgraphs for high risk exchanges incorrectly predicted

11 Conclusion

Our goal in this thesis was to suggest a methodology that financial institutions can use for risk detection when transacting in cryptocurrency markets. Overall, we were able to construct a model that performs well with respect to the priorities of a financial institution (achieving a high recall), though our modelling framework fails to sufficiently address requirements around scalability and explainability. We used Bitcoin transaction data to construct a novel address-level dataset and found that inputting this to a graph attention network (GAT) achieves significantly better recall scores than in previous literature, where models used transaction-level data. In addition, when we measure the performance of our model by value, we find that it is able to capture almost all high risk observations. Addressing other issues that financial institutions face in designing risk classification models, such as scalability and explainability, is more challenging. Since the transformed address-level dataset is much more computationally intensive for a model to train on and its performance on the test data is affected by the class imbalance inherent in the dataset, we suggest a 'pipeline' that an institution could implement to address scalability and performance issues. For this to become a reality, future work needs to find an approach to filter the data and discard a significant share of observations that are likely to be low risk. In terms of explainability, we present some anecdotal evidence based on summary statistics to consider what is driving the predictions of the model. However, we acknowledge that a more robust approach is needed for a financial institution to explain the predictions made by the model. We leave this to future work on this topic, though we suggest one way this could be approached.

Appendix A Model architecture

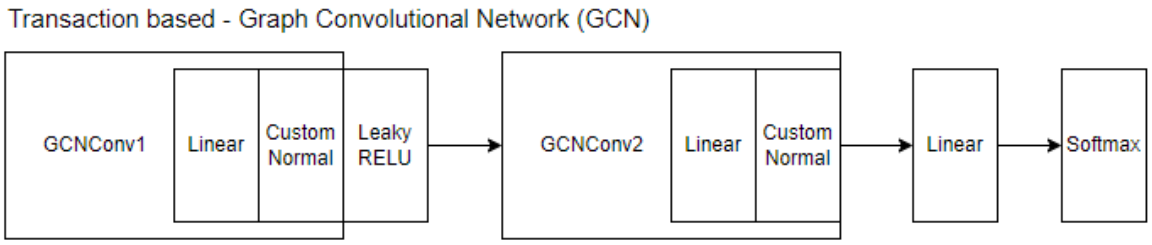

Figure 18: Graph convolutional network (GCN) architecture - transaction data as input

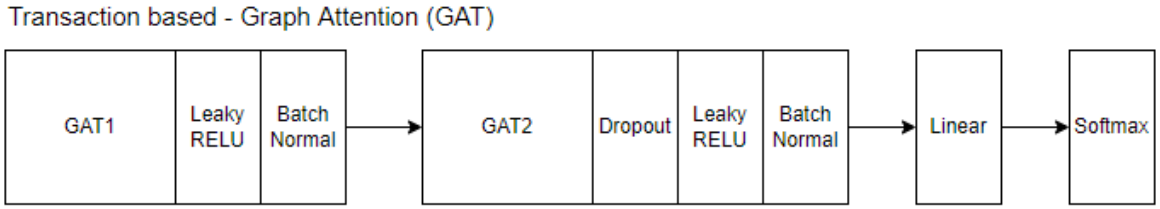

Figure 19: Graph attention network (GAT) architecture - transaction data as input

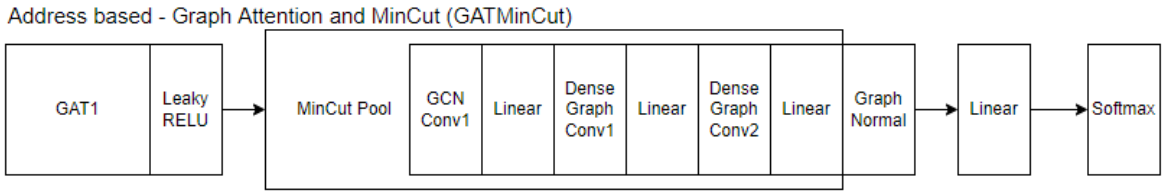

Figure 20: Graph attention network (GAT) architecture using MinCut pooling - address data as input

Appendix B Subgraphs for different Bitcoin address categories

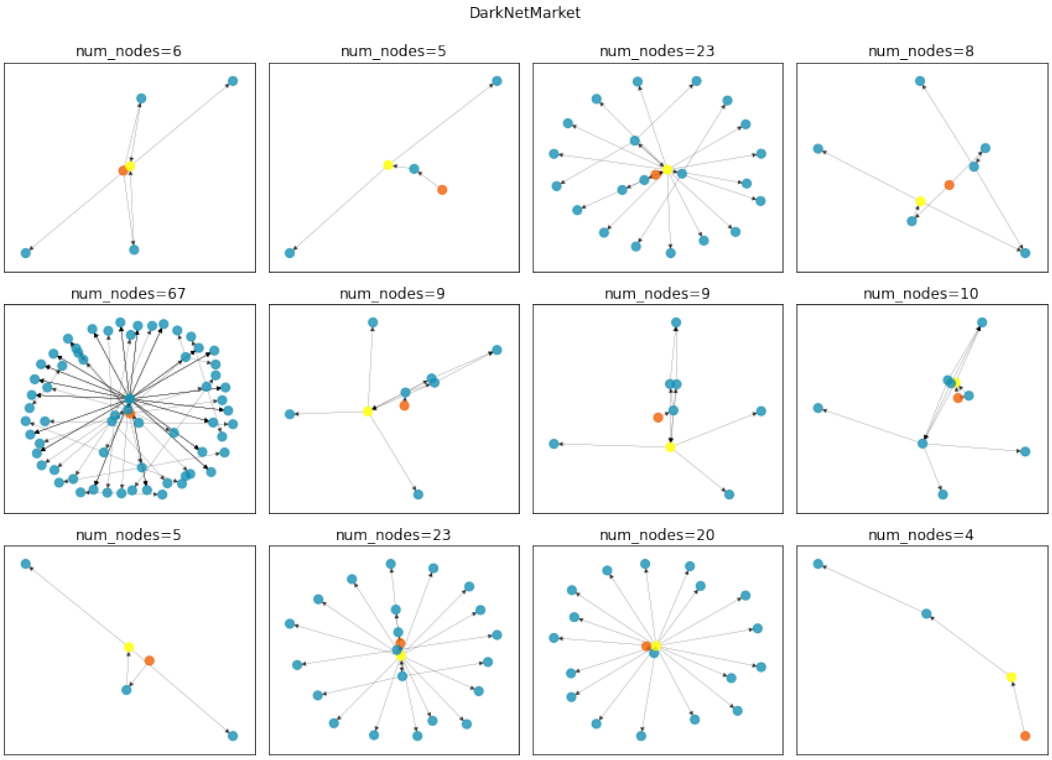

Figure 21: Subgraphs for darknet addresses

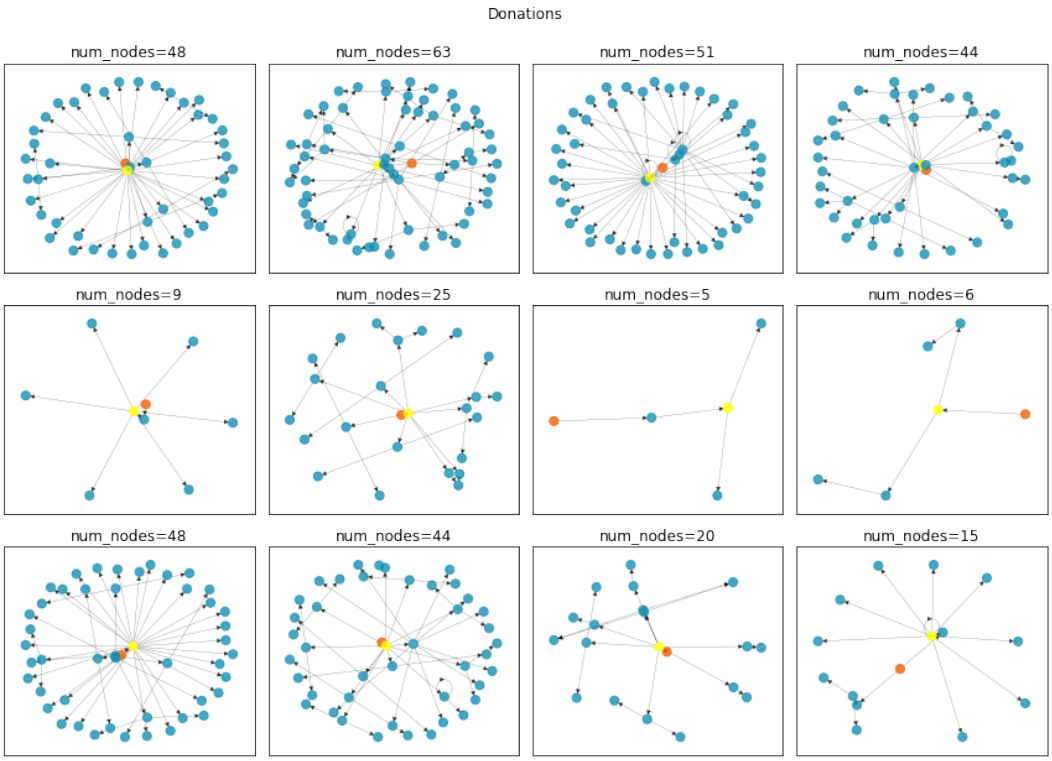

Figure 22: Subgraphs for donation addresses

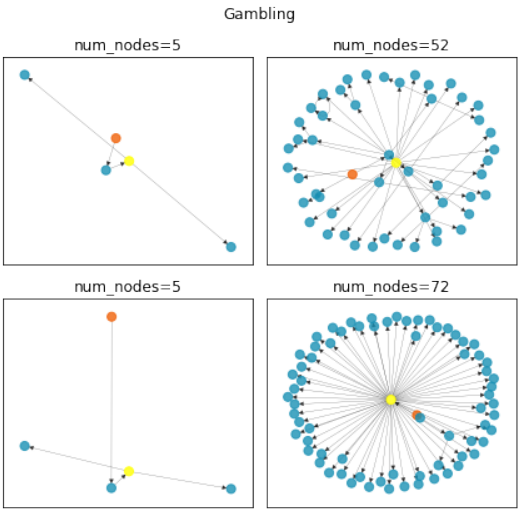

Figure 23: Subgraphs for gambling addresses

References

Agarwal, G. (2020). How to trace bitcoin transactions or address?

Alarab, I., Prakoonwit, S., and Nacer, M. I. (2020). Competence of graph convolutional networks for anti-money laundering in bitcoin blockchain. In *Proceedings of the 2020 5th International Conference on Machine Learning Technologies*, pages 23–27.

Arcos-Diaz, D. (2019). Graph convolutional networks for fraud detection of bitcoin transactions.

Back, A. et al. (2002). Hashcash-a denial of service counter-measure.

Benzik, A. (2019). Deanonymization of elliptic dataset transactions.

Bianchi, F. M., Grattarola, D., and Alippi, C. (2019). Mincut pooling in graph neural networks.

Brody, S., Alon, U., and Yahav, E. (2021). How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*.

Cai, T., Luo, S., Xu, K., He, D., Liu, T.-y., and Wang, L. (2021). Graphnorm: A principled approach to accelerating graph neural network training. In *International Conference on Machine Learning*, pages 1204–1215. PMLR.

Chainalysis (2022). Crypto crime trends for 2022.

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., and Sun, X. (2020a). Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3438–3445.

Chen, W., Guo, X., Chen, Z., Zheng, Z., and Lu, Y. (2020b). Phishing scam detection on ethereum: Towards financial security for blockchain ecosystem. In *IJCAI*, pages 4506–4512.

CoinMarketCap (2022). Total cryptocurrency market cap.

DeloitteUS (2021). The future of regulatory productivity.

Dernbach, S. (2018). Directed graph neural networks. *Unpublished manuscript*.

DoJ, U. (2020). Cryptocurrency enforcement framework.

Gasteiger, J., Bojchevski, A., and Günnemann, S. (2018). Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*.

Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

HorizenAcademy (2019). Utxo vs. account model.

Hu, Y., Seneviratne, S., Thilakarathna, K., Fukuda, K., and Seneviratne, A. (2019). Characterizing and detecting money laundering activities on the bitcoin network. *arXiv preprint arXiv:1912.12060*.

Kanezashi, H., Suzumura, T., Liu, X., and Hirofuchi, T. (2022). Ethereum fraud detection with heterogeneous graph neural networks. *arXiv preprint arXiv:2203.12363*.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kolesnikova, K., Mezentseva, O., and Mukatayev, T. (2021). Analysis of bitcoin transactions to detect illegal transactions using convolutional neural networks. In *2021 IEEE International Conference on Smart Information Systems and Technologies (SIST)*, pages 1–6. IEEE.

Lee, J., Lee, I., and Kang, J. (2019). Self-attention graph pooling. In *International conference on machine learning*, pages 3734–3743. PMLR.

Li, Liu, W. (2014). Fraud detection on bitcoin transaction graphs using graph convolutional networks.

Lin, D., Wu, J., Yuan, Q., and Zheng, Z. (2020a). Modeling and understanding ethereum transaction records via a complex network approach. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(11):2737–2741.

Lin, D., Wu, J., Yuan, Q., and Zheng, Z. (2020b). T-edge: Temporal weighted multidigraph embedding for ethereum transaction network analysis. *Frontiers in Physics*, 8:204.

Lou, Y., Zhang, Y., and Chen, S. (2020). Ponzi contracts detection based on improved convolutional neural network. In *2020 IEEE International Conference on Services Computing (SCC)*, pages 353–360. IEEE.

NewsBTC (2018). First instance of blockchain discovered in new york times circa 1995.

Ostapowicz, M. and Żbikowski, K. (2020). Detecting fraudulent accounts on blockchain: a supervised approach. In *International Conference on Web Information Systems Engineering*, pages 18–31. Springer.

Patel, V., Pan, L., and Rajasegarar, S. (2020). Graph deep learning based anomaly detection in ethereum blockchain network. In *International Conference on Network and System Security*, pages 132–148. Springer.

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710.

Poursafaei, F., Rabbany, R., and Zilic, Z. (2021). Sigtran: Signature vectors for detecting illicit activities in blockchain transaction networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 27–39. Springer.

Tan, R., Tan, Q., Zhang, P., and Li, Z. (2021). Graph neural network for ethereum fraud detection. In *2021 IEEE International Conference on Big Knowledge (ICBK)*, pages 78–85. IEEE.

Tang, J., Zhao, G., and Zou, B. (2022). Semi-supervised graph convolutional network for ethereum phishing scam recognition. In *Third International Conference on Electronics and Communication; Network and Computer Technology (ECNCT 2021)*, volume 12167, pages 369–375. SPIE.

Tian, H., Li, Y., Cai, Y., Shi, X., and Zheng, Z. (2021). Attention-based graph neural network for identifying illicit bitcoin addresses. In *International Conference on Blockchain and Trustworthy Systems*, pages 147–162. Springer.

Vasek, M. and Moore, T. (2015). ThereÃ¢â‚¬â„¢s no free lunch, even using bitcoin: Tracking the popularity and profits of virtual currency scams. In *International conference on financial cryptography and data security*, pages 44–61. Springer.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wang, J., Chen, P., Yu, S., and Xuan, Q. (2021). Tsgn: Transaction subgraph networks for identifying ethereum phishing accounts. In *International Conference on Blockchain and Trustworthy Systems*, pages 187–200. Springer.

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

Wu, J., Yuan, Q., Lin, D., You, W., Chen, W., Chen, C., and Zheng, Z. (2020). Who are the phishers? phishing scam detection on ethereum via network embedding. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Xia, P., Ni, Z., Xiao, H., Zhu, X., and Peng, P. (2021). A novel spatiotemporal prediction approach based on graph convolution neural networks and long short-term memory for money laundering fraud. *Arabian Journal for Science and Engineering*, pages 1–17.

Xu, M. (2020). Understanding graph embedding methods and their applications.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983.

Zhang, D., Chen, J., and Lu, X. (2021). Blockchain phishing scam detection via multi-channel graph classification. In *International Conference on Blockchain and Trustworthy Systems*, pages 241–256. Springer.