Anomaly Detection in Cryptocurrency Networks and Beyond

by

Farimah Ramezan Poursafaei

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering
McGill University
Montréal, Québec, Canada

August 2022

Abstract

Cryptocurrency networks that provide a new way of securing financial transactions have gained a surge of interest in recent years. However, recent studies reveal that blockchain networks are rampant with frauds and are prone to several privacy and security issues. The public availability of cryptocurrency transaction records provides an unprecedented opportunity for researchers to analyze cryptocurrency transactions. In particular, anomaly detection techniques are promising avenues for fighting illicit activities such as money laundering, terrorist financing, drug trafficking, scams, frauds, and many more.

In this dissertation, we address the challenges of detecting anomalous entities on cryptocurrency networks. Firstly, we introduce effective techniques for generating features for network entities that are directly devised from raw data and highlight the utility of those features in detecting illicit accounts on the Ethereum network. Next, we enrich the proposed method by expanding the feature set through the incorporation of graph-based features that embed the relational information of networks. This also enables us to generalize our methods to instances of cryptocurrency networks with different architectural models. Based on the success of our method in anomaly detection in cryptocurrency networks, we further generalize our model to encompass a generic temporal weighted multidigraph and show the state-of-the-art results for anomaly detection in other common domains including rating and social networks. In doing so, we also investigate the challenges of employing node classification techniques for anomaly detection, which is a common practice. Here, we discuss the importance of performance metrics and evaluation settings when interpreting the efficiency of different methods and tasks, which is often overlooked by the community. Finally, we shift our focus to examining the inherent challenges of learning on dynamic networks, which is an important emerging research field with applications in drug discovery, computational finance, social networks, etc. Here, we propose solutions for providing a more robust evaluation setup for dynamic graph learning methods.

The key contributions of this dissertation are twofold: First, we describe efficient techniques for detecting anomalies on cryptocurrency networks and generalize them to other real-world complex networks. Second, we focus on the temporal aspect of these networks and investigate how the dynamism of networks affects the downstream tasks and evaluation settings.

Abrégé

Les réseaux de crypto-monnaies qui offrent une nouvelle façon de sécuriser les transactions financières ont suscité un regain d'intérêt ces dernières années. Cependant, des études récentes révèlent que les réseaux de blockchain sont truffés de fraudes et sont sujets à plusieurs problèmes de confidentialité et de sécurité. La disponibilité publique des enregistrements des transactions en crypto-monnaies offre aux chercheurs une opportunité sans précédent d'analyser les transactions en crypto-monnaies. En particulier, les techniques de détection d'anomalies sont des pistes prometteuses pour lutter contre les activités illicites telles que le blanchiment d'argent, le financement du terrorisme, le trafic de drogue, les escroqueries, les fraudes, et bien d'autres encore.

Dans cette thèse, nous abordons les défis de la détection des entités anormales sur les réseaux de crypto-monnaies. Tout d'abord, nous introduisons des techniques efficaces pour générer des caractéristiques pour les entités du réseau qui sont directement conçues à partir de données brutes et nous soulignons l'utilité de ces caractéristiques pour détecter les comptes illicites sur le réseau Ethereum. Ensuite, nous enrichissons la méthode proposée en élargissant l'ensemble de caractéristiques par l'incorporation de caractéristiques basées sur les graphes qui intègrent les informations relationnelles des réseaux. Cela nous permet également de généraliser nos méthodes à des instances de réseaux de crypto-monnaies avec différents modèles architecturaux. Sur la base du succès de notre méthode de détection d'anomalies dans les réseaux de crypto-monnaies, nous généralisons notre modèle pour englober un multidigraphe pondéré temporel générique et montrons les résultats de l'état de l'art pour la détection d'anomalies dans d'autres domaines courants, notamment les réseaux d'évaluation et les réseaux sociaux. Ce faisant, nous étudions également les défis liés à l'utilisation de techniques de classification des nœuds pour la détection des anomalies, ce qui est une pratique courante. Nous discutons ici de l'importance des mesures de performance et des paramètres d'évaluation lors de l'interprétation de l'efficacité de différentes méthodes et tâches, ce qui est souvent négligé par la communauté. Enfin, nous nous concentrons sur l'examen des défis inhérents à l'apprentissage sur les réseaux dynamiques, qui est un important domaine de recherche émergent avec des applications dans la découverte de médicaments, la finance computationnelle, les réseaux sociaux,

etc. Ici, nous proposons des solutions pour fournir une configuration d'évaluation plus robuste pour les méthodes d'apprentissage de graphes dynamiques.

Les principales contributions de cette thèse sont de deux ordres : Premièrement, nous décrivons des techniques efficaces pour détecter les anomalies sur les réseaux de crypto-monnaies et nous les généralisons à d'autres réseaux complexes du monde réel. Deuxièmement, nous nous concentrons sur l'aspect temporel de ces réseaux et étudions comment le dynamisme des réseaux affecte les tâches en aval et les paramètres d'évaluation.

Acknowledgements

First and foremost, I am deeply grateful to my supervisors: Professor Zeljko Zilic and Professor Reihaneh Rabbany. I would like to express my special appreciation to Professor Zeljko Zilic for his insights and support. I am also extremely grateful to Professor Reihaneh Rabbany for her constant guidance, commitment, and great advice. I feel very lucky to have worked with her not only because of the numerous valuable work and life lessons that I learned from her, but also to have her as a role model and a great mentor. Our meetings were always very informative, a great source of inspiration, and had a huge impact on shaping my personal and academic vision.

I would also like to express my gratitude to my committee members: Prof. Gordon Roberts and Prof. Muthucumaru Maheswaran, for providing insightful comments and suggestions during my dissertation proposal and seminar.

I take this opportunity to thank my amazing teammates from the Complex Data Laboratory: Andy Huang, Kellin Perline, Aarash Feizi, Pratheeksha Nair, Michael Galkin, Albert Orozco Camacho, Devin Kreuzer, Jacob Danovitch, and Yifei Li. In particular, I would like to acknowledge the many enjoyable collaborations and research discussions that I had with Andy and Kellin during our weekly meetings and through developing our research projects. Moreover, I would like to thank my friends in Integrated Microsystems Laboratory: Amirhossein Shahshahani, Junchao Wang, Anastasios Alexandridis, and Ghassan Al-Sumaidaee.

On a personal note, I am thankful to my caring friends that made my life during the Ph.D. journey unforgettable: Nima, Arezou, Ehsan, Razie, Hossein, Tahereh, Shahin, Imman, Alireza, Saeed, Abbas, and many more. I have been tremendously lucky to have the chance to spend most of my free time with you experiencing many different adventurous and fun activities. Especially, I would like to thank Nima for being a real friend and a helpful person in every possible aspect, and Arezou for our joyful chats about nothing and everything. Cheers to many more years of friendship!

Finally, my deepest appreciation goes to my parents, Forooza and Reza, for their endless love and inspiration, and for always being truly supportive and motivating. I also want to thank my dear brother, Farshad, for his support and encouragement.

Contents

List of Figures

List of Tables

List of Abbreviations

List of Abbreviations:

Ether: ETH
DApp: Decentralized Applications
PoW: Proof-of-Work
PoS: Proof-of-Stake
UTXO: Unspent Transaction Output
EVM: Ethereum Virtual Machine
EOA: Externally Owned Account
ML: Machine Learning
AI: Artificial Intelligence
DL: Deep Learning
NN: Neural Networks
GNN: Graph Neural Networks
GCN: Graph Convolutional Neural Network
GAT: Graph Attention
DGNN: Dynamic Graph Neural Network
RNN: Recurrent Neural Networks
NS: Negative Sampling
RF: Random Forest
LR: Logistic Regression
SVM: Support Vector Machine
PCA: Principle Component Analysis
LOF: Local Outlier Factor
ISF: Isolation Forest
MLP: Multi-Layer Perceptron
AUROC: Area Under Receiver Operating Characteristic
AP: Average Precision
DTDG: Discrete Time Dynamic Graph
CTDG: Continuous Time Dynamic Graph

Chapter 1

Introduction

1.1 Motivation

The blockchain technology has emerged as one of the topics in the spotlight not only by revolutionizing finance via introducing cryptocurrency, but also by significantly changing other fields including healthcare [1, 2] and Internet of Things [3, 4]. The blockchain consists of a public ledger that records the history of all transactions and events permanently on a chain of blocks. Each block consists of a set of transactions, has a timestamp, is linked to its previous block, and is identified by its unique hash value. Before being appended to a block, transactions should be verified and signed via cryptographic hash functions. Participants of the blockchain are connected through a peer-to-peer network and each participant maintains a replication of the entire ledger.

The variety of the benefits that cryptocurrency networks offer has resulted in an unseen growing uptick in their usage. As of 2022, there are more than $10,000$ active cryptocurrencies being traded in online exchanges and the global cryptocurrency market capitalization has exceeded $\$1.9T$ [5, 6]. The total volume of cryptocurrency transactions has passed $\$15.8T$ in 2021, which has grown by %567 compared to 2020's total volume [7]. Although the legitimate cryptocurrency usage considerably surpasses the growth of criminal usage and the share of cryptocurrency illicit activities hits its all-time low, the total worth of illicit activities is still significant. For instance, the cryptocurrency crime trend reveals that illicit addresses on cryptocurrency networks have gained $\$14B$ through the year 2021, which is $\$6.2B$ more than the 2020's share [7]. Malicious utilization of the cryptocurrency increases the chance of governmental restriction and impedes the cryptocurrency's widespread adoption.

The core data structure of the blockchain consists of a chain of hash values associated with the blocks of transactions. The hash value of each block is computed based on the transactions of the block as well as the hash values of the previous blocks. Therefore, an update of a transaction in one block not only results in a change of the hash value of the block containing the transaction, but also changes the hash values of all succeeding blocks. This specific data structure makes it extremely difficult to delete or alter a blockchain transaction and makes the blockchain ledger tamper-resistant [8]. Although tamper-resistance increases the security of the blockchain by preventing some malicious activities such as double-spending, this feature becomes problematic when fraudulent transactions have been appended to the ledger [8]. Particularly, once an illicit transaction such as a scam or fraud, which can be made via stolen secret keys or miss operations, is appended to the blockchain ledger, it is infeasible to revise it due to the immutability of the blockchain ledger. Therefore, even if we successfully detect a malicious transaction that is already appended to the ledger, reverting its impact is not possible [9]. Thus, to prevent the damage, efficient countermeasures are required to detect abnormal events, illegal transactions, or predict suspicious users in advance.

Blockchain-based cryptocurrencies are not backed by any third parties or tangible assets, though they gain users' trust through disclosing their full transaction history [10]. Each cryptocurrency transaction in a blockchain network mainly consists of transferring a virtual value between a pair of virtual identities which are associated with blockchain addresses. Although extracting, transforming, and analyzing transaction histories are technically not easy tasks, the public availability of the blockchain ledgers provides unprecedented opportunities for researchers to examine the interactions in complex financial networks.

The importance of the network analysis has been proved in many diverse domains such as economic, social, or recommender systems. Similarly, for forensic analysis of cryptocurrency users' activities, network analysis is promising, since it facilitates the investigation of structural and temporal properties of the cryptocurrency networks. Considering that the complex network theory offers several techniques that are useful in the study of cryptocurrency networks, we can represent the flow of the blockchain transactions as a network. For instance, in Ethereum and its derivatives like EOS and Neo [11], the network nodes denote the account addresses, while the edges of the network represent the transactions which can involve cryptocurrency transfers, smart contract creations, or smart contract invocations. Utilizing the powerful quantitative measures of complex network analysis leads us to important observations and insights

about the cryptocurrency networks. For example, while most of the nodes in a cryptocurrency network have relatively small number of interactions, there exist several distinguishable nodes with a high degree that contribute the most towards the blockchain evolution. Exploring the interactions of the highly interactive nodes illustrates that it might be easier to realize the actual identity of these specific nodes since they are exposed to lower anonymity [12]. Thus, modelling and analyzing the cryptocurrency networks from a complex network perspective allows us to extract significant insights that are imperative for the successful exercise of other important tasks like anomaly detection. Additionally, like many other types of networks such as communication networks, biological networks, and social networks, it is quite conventional to represent the peer-to-peer distributed cryptocurrency networks as complex networks.

1.1.1 Network Data as Graphs

Networks can be modelled as graphs that consist of nodes and edges among the nodes, where the former corresponds to the network entities and the latter corresponds to the relations among entities. In a directed graph (*digraph*), a direction is associated with each edge, and in a *multidigraph*, there are multiple directed edges among pairs of nodes. The nodes or edges of the graph can have properties or specific labels.

Although the available massive cryptocurrency datasets contain affluent insights, exploiting the raw data for resolving challenging problems is not easy. For one thing, statistical or machine learning models cannot exploit the raw data, while manual interpretation of this huge data is impractical. For another thing, the network raw data involves laborious issues like being high-dimensional, or relational. To study the attributes and characteristics of real-world networks, we should take the relations of the network entities into accounts, since these relations contain important sources of information. Hence, a preliminary task for leveraging the large-scale network data is attaining effective *representations* or *features* from the network data to accommodate the relational complexities. Besides, rich and meaningful features are required to commensurate with the complexity of the datasets when tackling hard problems.

Like blockchain-based cryptocurrencies, networks are ubiquitous in various other domains such as rating or social platforms. The increasing prevalence of networks data emphasizes the need for efficient methods that exploit digital traces of the networks to improve our understanding and power of prediction of human behaviours. Leveraging

the network datasets and novel learning methods empower us to design more secure and efficient digital platforms. However, the main challenge lies in how we can utilize this heterogeneous data to synthesize meaningful insights for combating complicated crises such as illicit activities, frauds, money laundering, or human trafficking.

1.1.2 Graph Anomaly Detection

Despite many benefits of blockchain technology including security, immutability, distributed consensus, and tamper-resistance, certain illicit activities still threaten its security and it has been demonstrated that cryptocurrency networks are prone to a tremendous number of attacks and illicit activities [13, 14]. For instance, money laundering is performed by many malicious accounts that are regularly created, Ponzi schemes are utilized for stealing money from users, or malicious forks are generated for performing double-spending in the network [15]. Therefore, the importance of anomaly detection in cryptocurrency networks is considerable for the precise discovery of the vulnerabilities in a timely manner to prevent catastrophic consequences.

Essentially, anomalies are rare observations which are considerably deviating from others [16]. Anomaly detection aims to identify these rare observations by specifying considerable divergence from normal observations [17]. Normal observations are often generated by monitoring the typical activities of entities or network connections over a period. Anomaly detection has principal applications in preventing detrimental events such as financial frauds, social spams, or network intrusions [16]. Although anomalies rarely occur in real-world, for supporting downstream applications, it is critical to exploit the information associated with them. Particularly, studying the behaviours of fraudsters sheds light on useful evidence for the anti-fraud detection.

Conventional techniques of anomaly detection focus on representing real-world objects as feature vectors and identify the abnormal objects in the vector space [18]. Although these methods have been proved to be efficient in identifying outliers in more straightforward data formats, such as tabular data, they neglect the underlying complex relations among objects in network data. The relations among entities in real-world networks provide valuable complementary information that can significantly boost the performance of anomaly detection approaches [16].

Graph anomaly detection aims to distinguish anomalous graph objects (such as nodes, edges, sub-graph, etc.) and is often associated with increased difficulties due to the irregular structures, non-independence, and large scale of the graphs. Identifying

anomalies in graphs that comprise structural information raises a more complicated problem in non-Euclidean space, where abnormal observation can be associated with nodes (e.g., malicious users of a financial transaction network), edges, (e.g. malicious transactions in financial networks), or sub-graph (e.g., a specific malicious group of fraudsters in a financial network). Since the networks anomalies cannot be directly represented in Euclidean feature space, traditional anomaly detection cannot be directly employed for graph anomaly detection. Early works in graph anomaly detection often rely on engineering handcrafted features or utilizing statistical models by domain experts. However, this approach is not only very human-intensive, but also limits the possibility of detecting unknown anomalies. To tackle these challenges, recent studies search for the potential of adopting novel machine learning and deep learning solutions to detect anomalous graph objects [16]. By employing the valuable information from the structure and relations of the graphs, expressive representations can be generated in such a way that normal and abnormal objects can be easily separated. Therefore, detecting graph anomalies using machine learning methods is currently at the forefront of anomaly detection techniques in real-world networks [16].

1.1.3 Problem Studied

In this dissertation, we investigate the task of anomaly detection in cryptocurrency networks as an important emerging instance of financial networks and extend our graph-based anomaly detection methods to other kinds of real-world networks with diverse applications. Particularly, we study cryptocurrency transaction networks, rating platforms, and social networks aiming to acquire insights about the relations and attributes of the network's entities. We focus on malicious entities on these networks, and develop efficient representations for networks entities with the goal of detecting anomalies. The general theme of this dissertation is the investigation of cryptocurrency networks to detect traces of malicious entities and examine the possibility of extending the solutions to other kinds of networks. We start by studying the underlying technology of blockchain-based cryptocurrency networks and provide efficient approaches for detecting illicit users and transactions on these networks. Then, we aim to extend our solutions by examining novel learning methods on graphs that can improve the performance of anomaly detection techniques on real-world networks. Precise study of real-world networks reveals that employing models that are unaware of the temporal evolution of networks could result in negligence of certain important

properties of real dynamic networks. Therefore, we study various objectives and tasks, evaluation practices, and state-of-the-art (SOTA) methods on dynamic graphs.

1.2 Thesis Contributions

In this dissertation, we mainly focus on *anomaly detection in cryptocurrency networks* and the different ways that this problem can be generalized to other related tasks or networks. The dissertation starts by discussing illicit activities in networks, their scale, and their significance. Afterward, we focus on examining the research problem of detecting these activities, which is the focus of this research. Essentially, we start by investigating several approaches for detecting anomalous entities in cryptocurrency networks. First, we formulate the research problem as a classification task and propose a feature-based classification approach for detecting malicious addresses on the Ethereum network. Then, we consider different approaches for improving the proposed solution by reformulating the task as a node classification task in graphs which also extends to anomaly detection in other domains. We move towards graph-based methods for detecting illicit entities on Bitcoin or Ethereum networks. In addition, we extend the solutions and investigate how the same methods can be applied for anomaly detection in similar domains when we have a temporal graph. We also consider related tasks in the broader domain of temporal graphs. Particularly, we examine the application of node classification for anomaly detection and different setups that the node classification task can be applied. Finally, we present our investigations on a related task of link prediction in temporal dynamic graphs and how to improve the evaluation setup in this problem set. Investigating diverse aspects of this problem, we propose a simple baseline for dynamic link prediction as well as two negative sampling strategies that are useful in better evaluation of the dynamic link prediction task.

1.3 Contribution of Authors

A list of all publications is presented in Section 8. I would like to declare that I am the principal and first author among the co-authors. My contributions include developing the research ideas, collecting and processing the data, designing the experimental setups, implementing the models and baselines, analyzing the results, and writing the manuscripts. The author's supervisors, Prof. Zeljko Zilic and Prof. Rabbany, provided guidance, helpful discussion for developing the research ideas, comments, editorial

revisions, and research funding during the course of the process. My co-author in publication *#4*, Dr. Ghaith Bany Hamad, contributed by meaningful discussions during the development of the research as well as editing the papers. As for research *#5*, my co-author, Andy Huang, contributed with providing helpful discussion during developing of the research ideas, part of the data collection process, part of the implementations, and editing the manuscript. My other co-author in research *#5*, Kellin Perline, also helped with useful discussion thorough the project and editing the manuscript.

1.4 Overview of Thesis Organization

The focus of this dissertation is the study of the cryptocurrency networks aiming to detect malicious entities. We also investigate the applicability of the proposed solutions to other kinds of real-world networks. We introduce techniques for learning features of network entities and highlight the utility of such features in anomaly detection in large-scale networks in diverse domains such as blockchain-based cryptocurrency networks, rating platforms, and social networks. The key contributions of this dissertation are twofold: First, we explore the underlying characteristics of large-scale cryptocurrency networks whose data provides novel opportunities for studying behavioural patterns of financial network participants. Additionally, we focus on contrasting the behaviours of fraudsters and genuine network participants to obtain meaningful features that are useful in the anomaly detection task in real-world networks beyond cryptocurrency networks, such as rating or social networks. Second, we focus our attention on developing representations that can accommodate the relational complexity of networks data. In our study, we highlight the distinctness of dynamic and static graph models and illustrate the importance of considering the dynamic nature of real-world networks in developing representation learning methods together with the evaluation setup. In the remainder of this chapter, we provide a high-level summary of the organization of this dissertation.

The thesis starts with a broad exploration of different subcomponents of the task of anomaly detection on cryptocurrency networks and the viability of extending the proposed solutions to more general models of real-world networks. Specifically in Chapter 2, we start by investigating blockchain-based cryptocurrency networks and present the underlying technology, protocols, and different models of these networks. Afterward, an overview of the two of the most important cryptocurrencies, i.e. Bitcoin

and Ethereum, are introduced. This chapter continues by examining similar research on anomaly detection and related challenges such as dataset imbalance. Then, several network representation methods including network embedding methods and graph neural networks are studied. We also investigate the evaluation setting of the graph learning methods and dive into representation learning in dynamic networks.

Covering the related background on different components of the under-study problem, Chapter 3 presents a framework for detecting malicious entities on the Ethereum network utilizing features that are defined based on the attributes of the network participants [19]. We present a novel framework to identify malicious entities in the Ethereum blockchain network. The framework composes of an efficient method for extracting a set of features from the Ethereum blockchain data to represent the transactional behaviour of entities.

Since considerable information is embedded in relations of the network entities, in Chapter 4, we focus on developing signature vectors for detecting illicit activities in blockchain-based transaction networks that incorporate relational information [20]. According to the concurrent studies, cryptocurrency networks have evolved into multi-billion-dorllar havens for a variety of disputable financial activities, including phishing, Ponzi schemes, money laundering, and ransomware. We propose an efficient graph-based method, SigTran, for detecting illicit nodes on blockchain networks. SigTran first generates a graph based on the transaction records from the blockchain. Then, it represents the nodes based on their structural and transactional characteristics. These node representations *accurately* differentiate nodes involved in illicit activities. SigTran is *generic* and can be applied to records extracted from different networks.

Realizing the importance and challenges of the anomaly detection in real-world networks, we extend our research beyond the cryptocurrency networks to encompass other types of use-cases such as rating platforms or social networks in Chapter 5. In addition, we further investigate the temporal characteristics of real-world networks and incorporate dynamic attributes of the networks [21]. Study of the current literature on dynamic networks reveals that many real-world complex systems can be modelled by temporal networks. Representation learning on these networks often captures their dynamic evolution and is a first step for performing further analysis, e.g., node classification. Node classification is a fundamental task for graph analysis in general and in the context of temporal graphs, is often employed to categorize nodes based on their activity patterns. Analysis of existing real-world networks from different high-stake domains reveals that the rate of the malicious activities is on an uptick, resulting in catastrophic social or economic consequences. This strongly motivates designing

accurate node classification methods for temporal graphs. We propose TGBase, for node classification on weighted temporal networks. TGBase *efficiently* extracts key features to consider the structural characteristics of each node and its neighbourhood as well as the intensity and timestamp of the interactions among node pairs. These features *accurately* differentiate different classes of nodes, as shown on eight real-world benchmark datasets, outperforming multiple state-of-the-art deep/complex models.

Focusing on the anomaly detection task, we further investigate the viability of employing node classification methods for detecting anomalous entities, which is a common practice [22]. In Section 5.3, we explore the impact of graph-based techniques for detecting anomalous entities on real-world networks. Specifically, we focus on modelling the problem of detecting anomalous entities as a node classification task, and inspect the role of different approaches together with the evaluation setup and metrics to provide several useful recommendations for practical applications. We investigate different ways of handling the imbalance issue of the datasets which is a common problem when dealing with datasets containing anomalies, and demonstrate how a method that is agnostic to the dataset imbalance may show misleading performance.

Fascinated by the interesting yet challenging research opportunities on dynamic networks, in Chapter 6, we focus on examining dynamic network and learning methods that are specifically designed for them. We revisit current evaluation settings for the link prediction task on dynamic graphs. Using two novel visualization techniques for edge statistics in dynamic graphs, we observe that a large portion of edges in dynamic graphs naturally reoccurs over time, and recurrent patterns vary significantly across datasets. Based on these observations and motivated by real-world applications, we first propose two novel negative sampling strategies for the evaluation of link prediction in dynamic graphs. The performance of existing methods degrades significantly when the set of negative edges used during evaluation is chosen more selectively. This shows that it is necessary to conduct different negative sampling strategies beyond simple random sampling to fully understand the performance of a given method. Second, we proposed a simple baseline, EdgeBank, solely based on memorizing past edges.

Finally in Chapter 7, we conclude this dissertation with a summary of the contributions and outcomes of this research and provides several interesting yet challenging aspects that are open to investigate as future directions.

Chapter 2

Background and Related Work

2.1 Blockchain-based Crypotcurrency Networks

Blockchain technology becomes a game changer in many different application domains, especially financial applications [23]. One of the most important usecases of the blockchain technology is its adoption as an underlying technology in cryptocurrency networks. The newer generation of the blockchain technology provides an open source decentralized platform which enables a new paradigm of computing known as *Decentralized Applications (DApps)* which are running on top of the blockchains [23]. Essentially, the blockchain is an append-only ledger which keeps track of all the transactions and events happened in a network of mutually distrusted parties [24]. The integrity of the blockchain's records is guaranteed through a distributed consensus mechanism which makes it almost impossible to tamper the records of the ledger without being noticed by an entire network. Blockchains which offer a transparent and integrity protected data storage are usually managed by a peer-to-peer network. Thus, they are inherently resistant to the data modification. User computational devices, such as computers or mobile devices, are nodes of this network.

Blockchain technology provides a secure mechanism for gaining a verifiable and immutable sequence of records which are referred to as blocks and are publicly available to every node on the network. The blocks are chronologically ordered by their discrete timestamps. Blockchain is typically employed as a distributed ledger of transactions that is shared and synchronized across a peer-to-peer network. Based on a consensus protocol, participants of the blockchain network can reject or verify the record of the data waiting to be appended to the ledger. After accepting the records, they are

appended to the blockchain in chronological order of their verification and no more modification is possible. Generally in a blockchain-based cryptocurrency network, there are two mechanisms that empower users to administer the network without relying on any trusted parties: (I) a shared ledger of data, and (II) a consensus protocol [25]. Transactions are validated by special nodes, called *miners*, which have extensive computational capabilities and are able to append a new block to the ledger only if they can solve a *proof-of-work* puzzle [26]. Each miner that can successfully append a block to the ledger gains the fees associated with that block as a reward. The chance of winning the reward associated with each block (i.e.,the block's fee) is proportional to the computational power of the miners. The operational efficiency of the blockchain technology has increased by the widespread usage of cryptocurrencies such as Bitcoin [27] and Ethereum [11]. However, the application of the blockchain technology is not limited to cryptocurrencies and it has many different usecases such as tracking ownership of assets of high value, voting rights, healthcare applications, and many more.

The rise in the use of blockchain technology has a direct relation to the rise of security vulnerabilities such as phishing, hacking, and heist [24]. In addition, cyber-criminals are actively adopting the blockchain technology for their illegal activities due to its pseudonymity which lets them implement virtually untraceable scams [28]. The pseudonymity of the blockchain technology comes from the fact that participants of the network are able to make transactions without revealing their actual identities. In fact, participants use pseudonyms, specified as addresses, that can even be generated for every single transaction [28]. On one hand, there have been several attempts to de-anonymizing addresses on several blockchain-based cryptocurrency networks [29, 30, 31, 32]. On the other hand, a dual effort has been put into strengthening the anonymity of the blockchain [31, 33, 34, 35, 36] to reinforce the perception of facilitating illegitimate activities, while at the same time, making them hard to detect. Therefore, detecting any malignant behavior, which may be an identification of a malignant activity on the network, is of paramount importance.

Human analysis of the blockchain scams often requires a laborious phase of manual or semi-automated search on the web to collect addresses involved in malicious activities [37, 38, 39, 40, 41, 42, 43]. This phase is required for automatizing the analysis to quantify the impact of the malicious activities by inspecting the associated transactions and addresses on the blockchain. However, the problem that makes this approach intractable is that, in general, the addresses involved in malicious activities are not publicly available. Therefore, it is highly desirable to have tools that

automatically monitor the blockchain network for suspicious behaviors and identify the addresses associated with illicit activities. Considering the increasing volume of data being appended to the blockchain ledger, it is impractical to employ human analysis for the purpose of identification of suspicious transactions or entities [44]. Thus, automating the identification of malignant entities by Machine Learning (ML) techniques is essential.

Several widespread blockchain-based platforms that offer cryptocurrencies, such as Ethereum and Bitcoin, focus on providing broad flexibility, which results in a public dataset of transactions. It is widely believed that cryptocurrencies, like Ethereum, offer digital anonymity. It should be noted that Ethereum is anonymous in the sense that there is no direct way of associating user addresses with their actual identities. However, Ethereum addresses are uniquely identifiable on the network and the ownership of the addresses is not interchangeable. Thus, the activity records of the addresses can be tracked, aggregated, and analyzed. By effectively clustering the Ethereum address space, we can analyze user behavioral patterns. The results of these analyses can be later employed to predict the type of the owner of an unknown address. Moreover, such analysis can help in better understanding of network activities, enhance different strategies relying on the blockchain networks including trading strategies, and help in improving anti-money laundering practices [45].

The blockchain network has several characteristics that should be considered when investigating and analyzing its data. First, the volume of data available in the blockchain is very high. The investigation of this massive volume of data with manual or semi-automated traditional methods is practically impossible. Therefore, we should focus on the solutions that are capable of handling a high volume of data. Moreover, blockchain-based networks are very dynamic and constantly evolving, therefore updating and improving the analysis approaches should be possible within a certain period of time. In addition, the blockchain technology is the basis for transferring assets of high value, therefore its security is of paramount importance. Hence, the performance of the investigation and analysis methods is significantly important. Considering these inherent characteristics of the blockchain networks, adopting ML methods for the analysis of the blockchain data seems an appealing solution.

2.1.1 An Overview of Blockchain

The blockchain technology has received significant attention from academia as well as the industry because of its unprecedented innovations which provide the possibility for mutually distrusting parties to exchange financial data without the need for a trusted third party [17]. Blockchains are inherently resistant to the modification of the data, since they are managed by a peer-to-peer network, and they offer transparent and integrity-protected data storage. Blockchain provides a cryptographically secure mechanism for acquiring a sequence of records (known as *blocks*) that are publicly verifiable and immutable, while chronologically ordered by discrete timestamps. Every participant of the blockchain network (i.e. node) can observe the data and verify or reject it based on a consensus protocol. Upon the acceptance of a block of records, it is appended to the blockchain in chronological order of its verification and no more modification is feasible.

For block validation in blockchain networks, a distributed consensus mechanism is required due to the lack of a universal trusted party. Based on the key property used for achieving distributed consensus, there are different kinds of consensus protocols including:

• **Proof-of-Work**: a miner node (who is responsible to maintain the blockchain ledger) can succeed in having a block accepted only if the node can demonstrate that a predetermined amount of computational resources has been spent on that block.

• **Proof-of-Stake**: both a random selection procedure and the wealth or influence of the participating (miner) nodes are considered in reaching a consensus. The assumption is that the nodes with large stakes are interested in guaranteeing the blockchain integrity.

• **Proof-of-Elapsed-Time**: every potential miner node requests a secure random waiting time from a trusted execution environment (such as Intel's SGX) where the waiting time has been set in the computing platform. Each node waits for the assigned time to finish, and the first node that finishes will claim the validation leadership. Any trusted computing environment has a similar chance of being selected, therefore the number of resources contributed to the overall network specifies the probability for each node to be in control of the validation leadership [17].

The notion of the blockchain was first proposed implicitly by Nakamoto as the key underlying technology of the cryptocurrency known as Bitcoin [27]. Bitcoin uses a *transaction-centered* model which is known as unspent transaction outputs (UTXOs). Here, there is a distributed public ledger known as the blockchain in which the payment transactions between nodes of a peer-to-peer network are recorded. One important

difference between the traditional digital cash systems and the blockchain-based cryptocurrencies, such as Bitcoin and Ethereum, is the lack of a trusted third party (i.e. bank). Since Bitcoin only offers payment services, it is often referred to as *Blockchain 1.0*. The innovation of Bitcoin system is its consensus protocol; the nodes reach a consensus on the outcome after executing payment transactions.

The success of Bitcoin in addition to the need of providing applications more complex than just payments has inspired the notion of *smart contracts*. The concept of smart contracts leads to a new paradigm of DApps that run on top of blockchain technology [46]. Ethereum which uses an *account-centered* model (rather than the UTXO model of Bitcoin) offers its exclusive cryptocurrency known as Ether (ETH). Currently, Ethereum has become the de facto standard for DApps [23]. A total of 1.45 million smart contracts were created on Ethereum blockchain just in the first quarter of 2022 [47], and the market value of Ethereum is more than $166B as of July 2022 [5]. *Blockchain 2.0*, which goes much beyond the payment-centered Blockchain 1.0, has ushered due to the success of Ethereum.

Many high-profile attacks to the Ethereum systems have been observed in recent years [8, 46, 48, 49, 50]. Examples include the DAO attack [51] in 2016 where an attacker exploited a vulnerability in a smart contract code and stole approximately *$60M* [23]. In 2017, the vulnerability of the Parity wallet contract was exploited which caused the loss of *$31M* [52]. These attacks demonstrate the limited capabilities in securing the Ethereum system which is not a surprise, considering that smart contract programming is a new programming paradigm on top of blockchains.

2.1.2 An Overview of Bitcoin Network

Since its introduction in 2008, Bitcoin has emerged as the most well-known cryptocurrency among many competitors [14]. Bitcoin essentially exists in the form of sets of computer codes that virtually have monetary values. On one hand, Bitcoin offers several appealing characteristics including being fast, convenient, tax-free, and revolutionary. On the other hand, the security, reliability, and confidentiality of Bitcoin have been very controversial because of its disjoint form of the consolidated governance and law enforcement [14].

Fundamentally, blockchain provides an innovative decentralized consensus scheme for storing transactions, money transfers, and any other events in such a secure manner which eliminates the involvement of a trusted authority. In the Bitcoin peer-to-peer

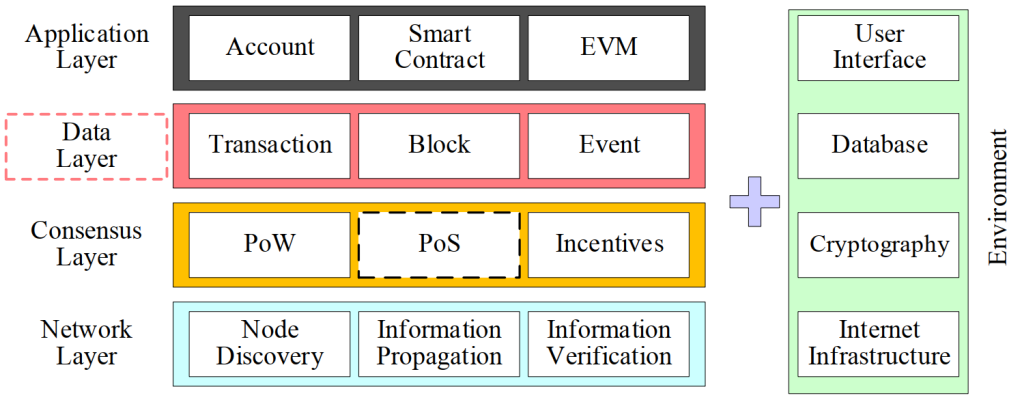


Figure 2.1: The Ethereum blockchain architecture [23].

network, each transaction is broadcasted to all network participants, and the integrity, authentication, and correctness of the transactions are verified by a special group of nodes, i.e. miners. A miner bundles several transactions that are waiting for verification into a single unit (i.e. block), validates the block and then advertises it across the whole network for claiming the rewards associated with the transactions of that block. Then, the majority of the miners on the network verify the block and upon the acceptance of the block, it is appended to a distributed public ledger known as the blockchain.

2.1.3 An Overview of Ethereum Network

As demonstrated in Fig. 2.1, the architecture of the Ethereum blockchain consists of four layers. The application layer is for the Ethereum users to execute smart contracts. The smart contracts are associated with Ethereum accounts and are executed on the Ethereum Virtual Machine (EVM). The data layer mainly contains the blockchain data structures. Every event happening on the network as well as all the transactions are recorded on the blocks which are appended to the public blockchain ledger. The consensus layer guarantees the consistent state of the blockchain among all nodes. Currently, Ethereum uses Proof-of-Work (PoW); however, it is planned to be replaced with Proof-of-Stake (PoS) [23]. The network layer manages the peer-to-peer networks of nodes in such a way that each node can get the updated state of the blockchain from some other active nodes. These four layers are served by the environment via different components: for interacting with different applications, a web user interface is employed; for storing the blockchain data, databases are adopted; cryptographic mechanisms are used for supporting the consensus protocols, and internet services are

exploited in the network layer. The functionalities of these different four layer are explained in more detail in the following subsections.

The Application Layer

Two types of accounts are supported in the Ethereum network: *externally owned accounts (EOAs)* and *contract accounts*. An EOA is used for keeping a user's funds. The funds are saved in *Wei* which is the smallest sub-denomination of ETH and is worth 10^{-18} ETH. An EOA is associated with a public key known as the public address. For accessing an EOA and controlling its fund, the corresponding private key is required which is used for authenticating the ownership of the EOA. In contrast, a contract account is associated with a smart contract which is a piece of executable bytecode and defines the desired business logic related to the smart contract of interest. Each account, EOA or contract, has a dynamic state defined by four different parameters: (a) Nonce: for an EOA account, this parameter keeps track of the number of transactions initiated by the owner of the EOA. For a contract account, this parameter tracks the number of contracts created by the contract account. (b) Balance: this parameter specifies the amount of Wei owned by the EOA or contract account. (c) Storage root: this parameter, which is only applicable to the contract accounts, is the hash of the root of the account's storage data structure tree. It records a contract's state variables associated with the corresponding piece of bytecode. (d) CodeHash: this parameter is only applicable to the contract account, and defines the hash value of a contract account's bytecode. In general, the state of the accounts on the blockchain specifies the state of the blockchain.

Essentially, the building blocks of DApps are smart contracts that are running on the Ethereum blockchain [23]. A DApp may consist of a user interface as its front end, while its back end is controlled by some smart contracts. There are plenty of different DApps running on top of Ethereum in diverse application domains including finance, governance, gambling, exchange, wallet applications, etc. [23]. For different purposes such as Initial Coin Offering (ICO) or exchange, some DApps may issue their cryptocurrencies which are known as *tokens*. The tokens that are based on Ethereum are considered a special kind of smart contract,e.g., ERC-20 [53]. Smart contracts are executed on EVM which has a stack-based architecture and is a quasi-Turing-complete machine. The execution of the commands of a smart contract on EVM is limited by the amount of gas provided as a reward incentive for the miners.

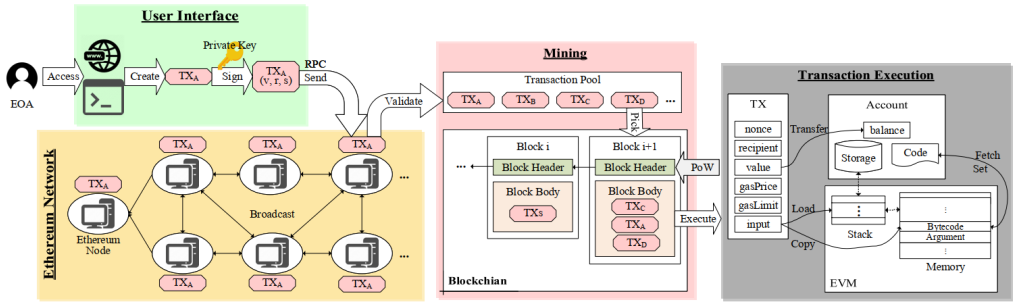


Figure 2.2: An Overview of the life cycle of an Ethereum transaction [23].

The Data Layer

Considering the Ethereum blockchain network, an interaction between two accounts is a transaction. A transaction is generally initiated by an EOA (i.e. sender) and is directed to another EOA or contract account (i.e. recipient). Each transaction contains different fields specifying the characteristics of the transactions. Some of the important fields of a transaction are as follows [23]: (a) Nonce: this field denotes a counter for tracking the total number of transactions that the sender has initiated. (b) Recipient: this field specifies the destination of a transaction that can be either an EOA or a contract account. (c) Value: this field specifies the amount of fund in Wei that should be transferred from the sender to the receiver if applicable; (d) Input: corresponding to the purpose of the transaction, this field specifies the bytecode or data needed for the transaction to be considered validly executed; (e) GasPrice: this field denotes the price of the unit of *gas* (i.e., reward) that the sender is willing to pay the miner who mines a block containing that transaction; (f) GasLimit: which specifies the maximum amount of gas that the sender is going to pay the winning miner who mines a block containing the considered transaction; (g) (v, r, s): which denotes the signature of the sender for verifying the transaction (the Elliptic Curve Digital Signature Algorithm (ECDSA) is considered as the main cryptographic algorithm for verification of the signatures). As a result of the execution of a transaction, the states of the involving accounts, as well as the state of the blockchain, are updated.

The general lifecycle of an Ethereum transaction is illustrated in Fig. 2.2. As shown in Fig. 2.2, a transaction lifecyle consists of different phases as follows [23]:

1. A sender generates a transaction and uses its private key to digitally sign it.
2. The sender can use a JSON-RPC to submit the transaction to an Ethereum node.
3. The Ethereum node validates the receipt of the transaction and broadcasts it to

the Ethereum peer-to-peer network.

4. Any miner who receives the transaction adds it to its transactions mining pool.

5. A miner chooses a sequence of transactions from its transactions mining pool, executes them, and generates a block of the selected transactions. Then, for appending the block to the blockchain ledger and updating the state of the blockchain, the miner should do the following tasks: if the transaction is a `money-transfer` transaction, the specified value is transferred from the sender account (i.e. EOA) to the recipient's EOA or contract account. If the transaction is a `contract-creation` transaction, a new contract account is created and it is associated with the bytecode provided by the input (which is a piece of bytecode). If the transaction is a `contract-invocation` transaction (in this case, the recipient is a callee contract, and the input identifies the callee function and the required arguments), the bytecode which is associated with the callee contract account will be loaded into the EVM.

6. The miner solves a PoW puzzle: the miner should find a random nonce such that the hash value of the block is smaller than a predefined threshold. The threshold controls the difficulty of creating the block.

7. After generating the block, the miner broadcasts it to the Ethereum network so that other nodes of the network can validate this block.

8. Finally after validating a block, the block is appended to the blockchain.

Each node on the Ethereum network stores the public blockchain ledger which consists of all validated blocks. Therefore, the history of all verified transactions is available to every node on the network.

The Consensus Layer

Creating a block on the Ethereum blockchain takes several seconds (specifically, Ethereum block mining takes about 12 to 15 seconds [54]). Hence, multiple miners could create valid blocks simultaneously, which requires a consensus about which new blocks to continue with and which one to reject. In Ethereum, the rejected blocks are called *uncle blocks* which are rewarded with a lower amount [55]. A modified version of the *Greedy Heaviest Observed Subtree (GHOST)* protocol derives the generation of an uncle block where the longest chain is selected as the canonical path [55]. On the Ethereum blockchain, not only the *regular* blocks on the main chain are rewarded, but also the stale blocks referred to as regular blocks are rewarded. However, there are different rewards to better advertising mining of the stale blocks. Upon mining of a block, the miner receives the gas fee associated with the transactions as well as some portion of the *static block reward* based

on the state of the blocks (i.e. whether the block is stale or on the main chain as well as its relation to a the main block).

The Network Layer

The Ethereum network consists of a structured peer-to-peer network in which each node stores a copy of the entire blockchain ledger. Nodes can discover other nodes and route the path on the network by means of a dynamic routing table that contains 160 buckets and each bucket has up to 16 entries including other nodes' Ethereum addresses, IP addresses, and UDP/TCP ports [23]. Special protocols are used for the discovery of different nodes and facilitating the exchange of the Ethereum blockchain information, such as blocks and transactions, among the clients.

The Environment

The environment in which the Ethereum blockchain is running naturally encompasses infrastructures to provide services across all different four layers [23]. For users to interact with the Ethereum blockchain, a web interface is devised. For the Ethereum nodes to store the blockchain data, a database is provided. For security purposes, cryptographic mechanisms are deployed, and finally for supporting networking and communication among Ethereum nodes, the internet infrastructure are considered.

2.1.4 Related Studies on Blockchain Transactions Analysis

There are different studies that investigate and analyze the transactional patterns of entities on blockchain-based cryptocurrency networks. We mainly focus on the related studies that employ ML methods for their analysis. There are different studies considering the adoption of ML solutions for anomaly detection in the blockchain technology [24, 56, 57, 58, 59]. These studies mainly consider Bitcoin which is the most widely used blockchain-based cryptocurrency. Monamo et. al. [56] apply clustering algorithms for labeling transactions committed to the Bitcoin blockchain ledger. The labels, which specify whether a transaction is normal or outlier, are further used to train and validate a supervised learning method to detect outlier instances in a set of unseen transactions. The main challenge of applying ML solutions in the blockchain environments is the scarcity of labeled data, which plays a pivotal role in the learning process [56, 57, 59].

Considering anomalies as abnormal or unlikely events, Pham and Lee [57] propose an anomaly detection solution for Bitcoin transactions which aims to detect suspicious

users and transactions. This study considers that illegal activities are essentially anomalous because the majority of the network participants are expected to behave logically. Therefore, anomalous behaviours can be considered as proxies for suspicious behaviours. The efficiency of the proposed approach is evaluated by checking whether the proposed solution is able to identify the transactions belonging to a group of known malicious transactions.

One important challenge of the previous studies in detecting Bitcoin anomalous transactions is the abundance of different assumptions in the process of evaluating the proposed approaches, which are made due to the fundamental problem of scarcity of the appropriate labeled data. For instance in [59], first, it is assumed that exactly *one* percent of all the transactions are anomalous. The authors also assume that the high popularity of Bitcoin causes more incentives for the attackers to leverage illicit activities to take advantage of the network; thus, there is a higher chance of detecting abnormal behaviors due to the abundance of illicit activities.

There are several studies focusing on detecting special cybersecurity illegal activities [28, 60, 61]. Bartoletti et al. [28] identify a special type of illicit investment known as *Ponzi scheme*. The presented approach consists of adopting data mining techniques for the automatic detection of Bitcoin Ponzi schemes using supervised learning algorithms. Although the idea of extracting the characteristics of special illegal activity and training a learning algorithm to predict such activity in unseen data is quite interesting, most the illegal activities are unprecedented and there is no way to retrieve the transactions related to a specific type of illegal activity to train the models accordingly. Similarly, Shaukat and Ribeiro [60] consider a special malicious software, known as *Ransomware*, where a cyber attacker uses the Bitcoin wallet to request ransom payment from a victim. This work does not primarily focus on the anomalous transactions on the blockchain, and only considers the adoption of the Bitcoin platform for transferring ransom payments.

In spite of the adoption of ML algorithms for anomaly detection in Bitcoin transactions, efficient learning solutions have been proposed for de-anonymizing [29, 30, 31, 32, 62, 63], as well as reinforcing the anonymity of addresses [31, 33, 34, 35, 36] on the blockchain network. For instance, Harlev et al. [62] exploit supervised learning algorithms for reducing the anonymity of the Bitcoin blockchain network, while using a pre-processed dataset (i.e., addresses of the transactions have been already clustered based on their behavioral patterns using manual analysis. Hence, different addresses that seem to be from a specific entity are clustered together and labeled according to their patterns). Using this pre-processed dataset, the proposed

approach aims to identify the category to which any of the unseen entities belongs. The objective of this study, similar to [63], is to reduce the anonymity of the Bitcoin blockchain network. However, the dataset provided to these studies is already labeled manually or through statistical analysis where the availability of such pre-processed data makes the proposed approach feasible.

2.1.5 Related Studies on Smart Contracts Security Exploration

In this section, we investigate the two main categories of the previous studies that analyze the security of the smart contracts and investigate existing vulnerabilities.

Static Analysis of Smart Contracts

Several studies investigate the vulnerabilities of smart contracts through analytical and static verifications [25, 64, 65]. Specifically, Atzei et al. [65] provide a survey of programming pitfalls in the Ethereum platform that may result in vulnerabilities of smart contracts. This study provides instances of real attacks that have exploited these vulnerabilities and resulted in considerable financial losses. Since the code of the smart contracts is fixed and it is impossible to change the codes after the creation and deployment of the contracts, if a security or semantic bug was unintentionally inserted into the code of the contract, there would be no way to debug the code to prevent an adversarial activity. Hence, it is often deemed that smart contracts are unpatchable and it is highly important to take necessary precautions when writing and developing smart contracts to deliver correctly implemented and secure contracts.

Several security problems that allow adversaries to manipulate the smart contract execution to gain benefits have been introduced in [25]. Luu et al. [25] believe that there exists a subtle gap in understanding of the distributed semantics of the underlying blockchain platform which results in many security issues. In this regard, they propose OYENTE [25] that is a symbolic execution tool for discovering specific types of potential security vulnerabilities in smart contracts before the actual deployment. This study investigates multiple security vulnerabilities in smart contracts which allow malignant participants to gain benefits. The vulnerabilities include transaction-ordering dependence, timestamp dependence, mishandled exceptions, and re-entrancy. Considering the characteristics of each of these vulnerabilities, OYENTE [25] tries to analyze Ethereum smart contracts by taking the bytecode of contracts as well as the current global state of the Ethereum as input, and then, it tries to identify the existence

of any of the considered vulnerabilities. Unfortunately, OYENTE is only able to inspect smart contracts for the vulnerabilities whose behaviors have been fully specified rather than a zero-day vulnerability.

It is worth noting that the previously proposed analyzer tools or techniques are mostly designed for specific types of attacks and they are impotent in confronting unseen illegal behavioral patterns. In contrast, the identified security breaches have been resolved in the newer version of the compilers, which made it impossible to generate and exploit the known attacks. Besides, an application-level solution cannot be extended to other cases, since it is based on the characteristics of the considered smart contract. A more practical approach consists in identifying attacks at the transaction-level. Needless to analyze the flow of the attack at the application-level, a solution at the transaction-level investigates the normal patterns of transactions and uses the acquired insight for predicting the authenticity of new transactions or users. Therefore, investigation at the transaction-level is not only independent of the considered smart contract but also is deemed to be scalable to diverse usecases.

Employing Machine Learning for Enhancing Smart Contracts Security

Chen et al. [61, 66] exploit the blockchain data for detecting Ponzi schemes that are implemented as smart contracts. They manually check the source code of several smart contracts, find out whether these contracts implement Ponzi schemes, and label the smart contracts accordingly. Then, they utilize the constructed dataset for training and evaluation of an ML classification model which is used for the detection of Ponzi scheme contracts. This work utilizes the source code of the smart contracts to generate the training dataset, which requires a high-level understanding of the functionality of target smart contracts. Moreover, it only considers Ponzi contracts, while the behavior of these contracts are fully characterized. Therefore, a new method to investigate the transaction-level patterns of smart contract participants without requiring any prior knowledge of the high-level behavior of the contracts is required.

Despite the adoption of ML algorithms for fraud detection in blockchain transactions, efficient learning solutions have been applied for other types of data analysis in blockchain networks. Harlev et al. [62] exploit supervised learning methods for reducing the anonymity of the Bitcoin blockchain network. The authors use a preprocessed dataset in which addresses of the transactions have been already clustered based on their behavioral patterns. Hence, different addresses that seem to be from a specific entity are clustered together and labeled according to their patterns. Using this

preprocessed dataset, the proposed approach aims to identify the category to which any of the unseen entities belongs. The objective of this study, similar to [63], is to reduce the anonymity of the Bitcoin blockchain. It should be noted that the dataset available for these research works has been already labeled manually or through statistical analysis, and the availability of such preprocessed data makes the proposed approaches feasible.

2.1.6 Categorization of Related Studies on Blockchain Analysis

A classification of the previous works together with their main objectives are presented in Table 2.1. In this classification, we consider different categories of the related studies. First, we consider the works that focus on the general concept of the blockchain-based cryptocurrency. Then, different groups of previous works with diverse objectives have been considered. Specifically, we consider the concept of anonymity and investigate several papers aiming to reduce the anonymity of the blockchain network. Another important category of the previous work is anomaly detection research; we investigate different research focusing on detecting anomalous behavior, fraud, or money laundering issues on the blockchain network. We also study the inherent vulnerabilities, attacks, and probable solutions of the blockchain-based cryptocurrencies, specifically Bitcoin and Ethereum. At the application-layer, we focus on different characteristics of smart contracts including general concepts and considerations, the standards for developing secure smart contracts, common and specific vulnerabilities of smart contracts, software engineering tools for developing, compiling, and verification of smart contracts, and finally adopting ML solutions for automatic classification of contracts. Apart from the blockchain-centered research, we also investigate several papers on credit card fraud detection and cybersecurity intrusion detection. Exploration of these two categories of research helps us better understand efficient solutions for better investigation of the blockchain data, since these categories share common concepts and effective solutions that can be employed for the problem of analysis of the blockchain data, in case a careful modification is applied.

2.2 Detecting Anomalous Entities in Networks

Cryptocurrency Networks. The increasingly huge amount of data being appended to the blockchain ledger makes the manual analysis of the transactions impossible.

Table 2.1: An overview of several previous studies on the blockchain analysis.

Category	Main Focus	Network	ML	Reference
General concept	Introduce Bitcoin	Bitcoin	✗	[27]
	Discussion	Bitcoin	✗	[14]
		Bitcoin	✗	[9]
Anonymity	De-anonymizing	Bitcoin	✗	[30]
		Bitcoin	✓	[62]
Anomaly detection	Anomalous Behavior	Bitcoin	✓	[67]
		Bitcoin	✓	[56]
		Bitcoin	✓	[59]
		Bitcoin	✓	[57]
		Ethereum	✓	[68]
		Ethereum	✓	[58]
		Bitcoin	✓	[44]
		Bitcoin	✓	[24]
	Anti-money laundering	Bitcoin	✓	[45]
Vulnerabilities	Discussion	N.A.	✗	[64]
	Vulnerability detection	Bitcoin	✓	[28]
Smart contracts	Setting standards	Ethereum	✗	[69]
	Common vulnerabilities	Ethereum	✗	[25]
		Ethereum	✓	[70]
	Comprehensive discussion	Ethereum	✗	[65]
		Ethereum	✗	[71]
		Ethereum	✗	[23]
		Ethereum	✗	[72]
	A specific vulnerability	Ethereum	✓	[66]
		Ethereum	✓	[61]
		Ethereum	✗	[23]
	Software engineering tools	Ethereum	✗	[73]
	Labeling of smart contracts	Ethereum	✓	[74]
Fraud detection	Credit card	Financial	✓	[75]
		Financial	✓	[76]
		Financial	✓	[77]
Intrusion detection	Cybersecurity	N.A.	✗	[78]
	IDSs and blockchain	N.A.	✗	[17]

Multiple works propose different ML techniques for detection of the illicit entities on cryptocurrency networks [24, 45, 62, 79, 80, 81]. Specifically, [24, 45, 62] investigate supervised detection methods for de-anonymizing and classifying illegitimate activities on the Bitcoin network. In parallel, Farrugia et al. [80] focus on examining the transaction histories on the Ethereum aiming to detect illicit accounts through a supervised classification approach.

Lorenz et al. [79] address the problem of tracking money laundering activities on the Bitcoin network. Concentrating on the scarcity of the labeled data on a crypocurrency network such as Bitcoin, Lorenz et al. [79] argue against the unsupervised anomaly detection methods for the detection of illicit patterns on the Bitcoin transaction network. Instead, they propose an active learning solution as effective as its supervised

counterpart using a relatively small labeled dataset.

Previous works often focus on analyzing the transactions on the cryptocurrency networks with *platform-dependent* features, e.g. works in [56, 79, 80, 82]. There are also several *task-dependent* studies investigating a specific type of illicit activities, for example [28, 61] focus on Ponzi schemes, which are illicit investments.

Graph-based anomaly detection approaches have emerged recently as a promising solution for analyzing the growing data of the blockchain ledger, for both Bitcoin [83], and Ethereum [81, 84, 85, 86, 87]. In particular, Wu et al. [85] model the transaction network of Ethereum as a multi-edge directed graph where edges represent transactions among Ethereum public addresses, and are assigned weights based on the amount of transactions and also a timestamp. They propose *trans2vec* which is based on adopting a random walk-based network embedding method and is specifically designed for the Ethereum network. Weber et al. [45] model the Bitcoin transaction network as a directed acyclic graph where the nodes represent the transactions and the edges illustrate the flow of cryptocurrency among nodes. They construct a set of features for the transactions of the network based on the publicly available information. Then, they apply Graph Convolutional Networks to detect illicit nodes. For node features, they consider local information (e.g. transactions fee, average Bitcoin sent/received), and other aggregated features representing characteristics of the neighborhood of the nodes.

In addition, there is an increasing surge of interest in employing machine learning methods for the study and investigation of blockchain-based cryptocurrency networks in recent years. Considering the abundance and public availability of the blockchain-based cryptocurrency data, various tasks have become viable using machine learning algorithms. These tasks span from general knowledge discovery in the cryptocurrency networks [10, 54, 87, 88, 88, 89, 90, 91, 92] to other challenging tasks [93] including fraud detection [94, 95, 96], rate prediction on cryptocurrency exchange [97], cryptocurrency price analysis [98, 99, 100, 101, 102, 103, 104, 105], security threats analysis [49, 50, 106, 107, 108, 109, 110, 111, 112, 113], and scalability analysis [114].

2.2.1 Anomaly Detection via Node Classification

An important supervised task on graphs is the node classification in which the objective is to gain node representations that can be utilized for accurate prediction of node labels. Node classification is employed by many related research for anomaly detection in different applications. In this regard, there are several methods that leverage

heuristic approaches for generating task-dependent node representations tailored to a special application. The generated features are then exploited in a downstream task of node classification. Depending on the characteristics of the datasets or usecases, the node classification task may be applied aiming to detect anomalous instances; examples of which include credit card fraud detection [76, 115, 116, 117], detecting addresses associated to phishing activities in cryptocurrency transaction networks [19, 20, 118, 119], and prediction of fraudsters in rating platforms [120, 121].

Depending on the platforms and the underlying networks, different sets of features can be extracted. For instance in rating platforms, timestamp [122] or reviews [123] associated with ratings are exploited to derive meaningful features. Specifically, *REV2* [120] focuses on predicting fraudulent users on rating platforms. It considers a rating network as a signed graph and generate node representation vectors through an iterative process while incorporating behavioral attributes of the users. In cryptocurrency networks, [80, 124] produce features for the Ethereum accounts based on their transaction histories, while [45, 125, 126, 127, 128] consider the Bitcoin transactions network and aim to detect instances of anomalous activities such as phishing, money laundering, and ransomware attacks.

More recently, Graph Neural Networks (GNNs) have been widely adopted for detecting anomalous entities in graph-structured data [16, 129, 130, 131, 132, 133]. In particular, Goodge et al. [134] unify local outlier methods and show that these methods are special cases of GNNs' message passing framework. Liu et al. [133] focus on issue of heavily unbalanced distribution of the node labels in anomaly detection applications and propose a solution for imbalanced supervised learning on graphs. Deng and Hooi [135] combine GNNs and a structure learning approach aiming to provide explainability for anomaly detection.

2.2.2 Imbalanced Classification

Class imbalance problem is one of the important issues in data mining and it refers to the situation where one of the classes has significantly less number of instances than the others [136]. While in many cases, like anomaly detection scenarios, instances of the minority class are very important, most of the algorithms mainly focus on classification of the majority classes and neglect minority samples [136]. Traditionally, there are three flavors of methods intending to tackle the class imbalance problem namely data-level, algorithm-level, and hybrid [136]. Data-level methods adjust the class imbalance by

under- or over-sampling [137]. When applying under-sampling, samples from majority classes are discarded aiming to make balanced classes, which causes information loss. On the other hand, over-sampling involves replication of minority samples for reducing class imbalance, though it comes at the price of over-fitting.This problem happens as no additional information is introduced by the replicated samples and one solution for this issue has been proposed as *SMOTE* [138]. In contrast to the data-level methods, algorithm-level methods assign different mis-classificaion penalties to different classes aiming to increase the importance of the minority samples [139]. Finally, the hybrid methods combines ideas from data-level and algorithm-level approaches to propose countermeasures against class imbalance issue [140, 141].

2.3 Graph Representation Learning

Generally, graph representation learning has become commonplace in network analysis with superior performance in a wide range of real-world tasks including the detection of illicit entities in a network. Graph representation learning includes several diverse categories. Some of the relevant techniques are introduced in the following sections.

2.3.1 Network Embedding

These methods aim at developing a mapping function from a discrete graph to a continuous domain [142, 143]. The goal is to learn a vector representation for each node such that important global or local properties are preserved [144, 145, 146]. Shallow embedding methods are similar to lookup tables where the vector representation of a node is found by its ID [142, 147]. A popular category of these methods is the random walk-based approach where the embeddings are learned in an unsupervised manner [145]; examples include *node2vec* [148], and *RiWalk* [149]. In random walk-based approaches, for each node of the network, several random walks with a fixed length are generated. The nodes in these walks are then considered as words of sentences where their representations are learned through natural language processing techniques (e.g., by skip-gram architecture [150]). As pioneering methods, *deepwalk* [151] and node2vec [148] employ random walks to capture the structure of the given graph which are fed into the skip-gram architecture [150] to generate embeddings that place similar nodes close to each other. With the biased random walks, node2vec embeddings can interpolate between embeddings based on the community structure and structural roles.

More recently, *RiWalk* [149] proposes to relabel nodes based on their structural roles before generating the random walks to further improve the embeddings. Specifically, RiWalk first assigns new labels to the nodes based on their structural roles. After that, it employs the random walk generation and skip-gram model for extracting final embeddings. For a supervised problem such as node classification, these methods first learn the embeddings in an unsupervised way and then train a supervised classifier in the embedding space. The shallow unsupervised embeddings models are outperformed by more powerful Graph Neural Networks (GNNs) specifically for node classification where it seems better to learn the task in an end-to-end fashion.

2.3.2 Graph Neural Networks (GNNs)

These methods mainly employ message passing procedures over the graph to generate node representations by feature smoothing over the local neighborhoods, where the representations are initialized with the explicit node features when the features are available [152, 153, 154, 155, 156, 157]. *Graph Convolutional Network (GCN)* [154] is a pioneering model that learns node representations for the semi-supervised node classification by deriving the importance of each neighbor in the aggregate function directed from a normalized adjacency matrix. *Graph Attention (GAT)* [156] is an attention-based variation of GCN where at every layer, it learns the importance of neighbor nodes instead of deriving it. *GraphSAGE* [155] reduces the computational complexity of GCN by sampling the neighbors. Xu et al. [158] focus on understanding the representational properties and limitations of GNNs and propose a theoretical framework for analyzing the expressive power of GNNs in terms of capturing different graph structures. They also propose Graph Isomorphism Network (GIN) and theoretically and empirically show that GIN has a high representational power [158].

There is a surge interest in the study and application of GNNs. In particular, there are several studies survey the extension of deep learning for graph-structured data [159, 160, 161, 162, 163, 164, 165, 166, 167, 168]. There are several surveys investigating applications of GNNs in different domains [163, 169] which includes recommendation systems [170, 171, 172], traffic forecasting [173, 174, 175], and natural language processing [176, 177]. Some researchers study diverse tasks defined on GNNs. These studies span in different domains such as the study of graphs with heterophily [178], node classification task [179], financial applications [180], expressivity of GNNs [181], explainability of GNNs [182], compute vision [183, 184], and drug discovery [185, 186, 187].

2.3.3 Benchmarking Graph Learning Methods

Among the key factors for evaluating the performance of different methods are the availability of benchmark datasets and comprehensive benchmarking frameworks. There are various benchmarks for general graph mining [188], graph representation learning [189], graph contrastive learning [190], graph robustness evaluation [191], graph-level anomaly detection [191], as well as benchmarks for time-series outlier detection [192], and outlier detection for tabular data [193].

Several studies identify several issues in the evaluation of existing GNN models. Focusing on static graphs, Dwivedi et al. [194] identify issues with a comparative evaluation due to inconsistent experimental settings. Shchur et al. [195] show that reusing the same train-test split in many different works has led to overfitting and using different splits of the data could result in a different ranking of the methods. Similarly, Errica et al. [196] highlight issues with GNN evaluations, while focusing more on reproducibility. Hu et al. [188] aim to facilitate reproducibility and scalability of graph learning tasks by providing a diverse set of datasets, unified evaluation protocols, metrics, and data splits. Focusing on heterogeneous GNNs, Lv et al. [197] find that improper setting leads to underestimation of simple homogeneous GNN methods.

Evaluating Link Prediction: Yang et al. [198] examine the importance of the selected metric for reporting the performance of various methods and argue that metrics based on threshold curves such as AU-ROC (Area Under the ROC curve) are more suitable. Regarding dynamic graphs, Junuthula et al. [199] differentiate dynamic and static link prediction by the consideration of edge insertion or deletion. Junuthula et al. [200] consider the problem of incorporating information from friendship networks into predicting future links in social interaction domains. Haghani and Keyvanpour [201] provide a comprehensive review of link prediction methods for social networks and categorize the link prediction task into two groups: missing link prediction, and future link prediction.

2.3.4 Negative Sampling (NS) of Edges in Graphs

Although the majority of the research in sampling and network embedding investigates criteria to efficiently sample positive edges, there are several studies examining the role of NS on performance. Yang et al. [202] argue that NS is as important as positive sampling in graph representation learning. For static link prediction, the most common method is to sample negative edges at random [148, 203, 204]. Alternatively, the sampling can be

based on connecting nodes with specific properties (e.g., a sufficiently large degree) [205], or it can be based on a particular geodesic distance [206, 207]. For example, Maruf and Karpatne [208] sample negative edges based on the shortest path of node-pairs, which results in maximization of similarity of nearby nodes as well as dissimilarity of distant nodes in the embedding space. Kotnis and Nastase [209] provide an empirical study of the impact of different NS strategies during training on the learned representations of various methods in knowledge graphs.

2.3.5 Dynamic Graph Neural Networks

A dynamic network is considered a network where edges and nodes appear and/or disappear, or the attributes of the nodes or edges change over time [210]. A Dynamic Graph Neural Network (DGNN) is a deep learning-based architecture that encodes both temporal and structural attributes of dynamic networks and mainly holds two properties. First, it has a neural network architecture that encodes a dynamic network. Second, neighboring node features are aggregated as part of the neural network architecture. DGNNs often make use of graph neural networks (GNNs) and time series modules (like Recurrent Neural Networks (RNN)) for encoding structural and temporal patterns, respectively.

Depending on the temporal granularity, a dynamic network can be represented in one of the following ways [210]: (a) static (no temporal information), (b) edge-weighted (the temporal information is included as labels on the edges and/or nodes of a static network), (c) discrete (multiple snapshots, represented in discrete time intervals), (d) continuous (no temporal aggregation). Dynamic graphs can be investigated from several perspectives including *temporal granularity*, *node dynamics*, and *link duration*. From the temporal perspective, we can have *static*, *edge-weighted*, *discrete*, and *continuous* networks, where only the last two categories are considered dynamic networks [210]. From a node dynamics perspective, we group networks into *static*, *dynamic*, or *growing* networks. From a link duration perspective, we have a spectrum including *interaction* networks, *temporal* networks, *evolving* networks, and *strictly evolving* networks.

DGNNs combine two modules: deep time series encoding and aggregation of neighboring nodes. In the discrete version, most often a GNN is combined with an RNN. A discrete DGNN considers a dynamic network snapshot by snapshot and encodes each snapshot all at once. This is similar to how static GNN encodes static networks. However, in the continuous version, baseline GNNs can not be adopted

directly because they need a static graph, thus the node aggregation should be modified. A continuous DGNN goes over the dynamic network edge by edge, thus no notion of the size of any snapshot is required. The important property of all DGNNs is their intention to capture temporal as well as structural patterns and encode them into embeddings [210].

Several recent studies examine diverse approaches for learning the representations of dynamic graphs [142, 210, 211, 212, 213, 214, 215, 216, 217]. For modeling dynamic graphs, several methods employ RNNs as a common choice for extending the sequence models to temporal graphs [142]. *JODIE* [218] focuses on the temporal interactions of the users and items in social networks and learns the node embeddings through coupled RNNs. *DyRep* [219] is a model that updates the node representations upon observing a new event in the network via its custom RNN. *TGAT* [220] propagates messages from sampled neighbors of a specific node similar to GNN methods. The sampling strategy of TGAT requires saving historical neighbors, which may hinder the method from being adopted in online learning. *TGN* [221] combines graph-based and memory operators for presenting an efficient and generic framework for learning on temporal graphs. Rossi et al. [221] demonstrate that TGN is a generalization of several models for learning on temporal graphs which are presented as a sequence of events.

Recently there is an increasing number of studies targeting temporal networks. Kazemi et al. [142] present a survey of advances in representation learning on dynamic graphs. They mainly review two categories of methods. First, they examine methods that capture the essence of nodes and edges of evolving graphs by producing time-dependent representations. Second, they introduce various methods that use representations for distinct downstream tasks such as node classification, event prediction, or link prediction. More recently, Skardinga et al. [210] concentrate on recent studies on DGNNs and provide a detailed terminology of dynamic networks. Skardinga et al. [210] and Kazemi et al. [142] both argue that modeling dynamic graphs with continuous representations has a higher potential, since it offers a superior temporal granularity. With that in mind, in our experiments we further investigate five recent models of this type:

- **JODIE [218]**: focuses on bipartite networks of instantaneous user-item interactions. JODIE has an update operation and a projection operation. The former utilizes two coupled RNNs to recursively update the representation of the users and items. The latter predicts the future representation of a node, while considering the elapsed time since its last interaction.

- **DyRep [219]**: has a custom RNN that updates node representations upon observation of a new edge. For obtaining the neighbor weights at each time, DyRep uses a temporal attention mechanism that is parameterized by the recurrent architecture.

- **TGAT [220]**: aggregates features of temporal-topological neighborhood and temporal interactions of a dynamic network. The proposed TGAT layer employs a modified self-attention mechanism as its building block where the positional encoding module is replaced by a functional time encoding.

- **TGN [221]**: consists of five main modules: (1) *memory*: containing each node's history and used to store long-term dependencies, (2) *message function*: for updating the memory of each node based on the messages that are generated upon observation of an event, (3) *message aggregator*: aggregating several messages involving a single node, (4) *memory updater*: responsible for updating the memory of a node according to the aggregated messages, and (5) *embedding*: generating the representations of the nodes using the node's memory as well as the node and edge features. Similar to TGAT, TGN also utilizes time encoding for effectively capturing the inter-event temporal information.

- **CAWN [222]**: generates several Causal Anonymous Walks (CAWs) for each node, and uses these CAWs to generate relative node identities. The identities together with the encoded elapsed time are used for encoding the CAWs by an RNN. Finally, the encodings of several CAWs are aggregated and fed to an MLP for predicting the probability of a link between two nodes.

As shown in the experiments section, these methods can often achieve very high performance for the link prediction tasks on dynamic graphs. This hinders researchers' ability to evaluate if new models are superior. Also, it exaggerates the efficacy of current models on real-world tasks. Hence, we further examine the evaluation procedure, from the perspective of both benchmark datasets and negative sampling.

Chapter 3

Detecting Illicit Accounts on a Cryptocurrency Network

In this chapter, we present our work on detecting malicious Ethereum entities by formulating the problem as a classification task. For detecting malicious entities on the Ethereum cryptocurrency network, we need labeled data. There was no labeled dataset available at the time of conducting this research, since this task was mainly tackled by unsupervised methods. Here, we explain how we curated a labeled dataset of malicious accounts and analyzed the malicious activities on the Ethereum blockchain network.

3.1 Introduction

For safeguarding online financial systems, a critical role is carried out by anti-money laundering regulations. However, applying such regulations increases costs. Blockchain-based cryptocurrencies enable the investigation of network-level interaction and forensic analysis due to the public availability of their ledger [223]. However, analysis and investigation of the interactions on the blockchain network are laborious tasks due to its dynamic nature, the large amount of data, and a high level of anonymity. Particularly, blockchain-based cryptocurrencies allow the illegitimate parties to easily hide in plain sight by providing a high degree of anonymity. In fact, current anti-money laundering methods are doing a poor job of stopping money laundering efforts [45]. Therefore, developing efficient solution for detection of suspicious users or activities is necessary for online transactions.

Widespread adoption of cryptocurrencies such as Bitcoin and Ethereum paved the way

for further investigation on emerging technological and economical issues [9]. Different scenarios of misuse and abuse resulted in considerable financial losses in addition to the unreliability of Bitcoin and Ethereum [9, 17, 65]. For instance, the anonymity of the blockchain-based cryptocurrencies is employed by the criminals to compromise the network security by running illegitimate activities such as ransomware attacks, or being involved in the dark business of human trafficking, drug cartel, and exchange of illegal goods and services [28, 45].

Blockchain-based cryptocurrencies impose new security vulnerabilities resulting from illegitimate activities related to phishing, hacking, heists, and scams. Therefore, detecting any malicious user or behavior at the initial state is of paramount importance for preventing further disastrous failures. Although different methods were proposed to detect malicious users or transactions through the inspection of the transactional records of entities on the blockchain networks, they mostly relied on the laborious manual or semi-automated analysis process that have been shown to be ineffective [224, 225]. Hence, a goal is to attain automatic analysis tools that identify the public addresses of the entities associated with malicious activities for improving the blockchain networks security. Furthermore, it is often impractical to employ human analysis for identifying the suspicious addresses on the blockchain network due to the increasingly high volume of data appending to the ledger [44].

Although several studies dealing with the problem of detecting malicious entities in the Bitcoin network exist [45, 56, 226], they are not directly applicable to Ethereum due to the fundamental differences between Bitcoin and Ethereum [227]. While Bitcoin is based upon *UTXOs (Unspend Transaction Outputs)* model, where the transactions constitute building blocks of the blockchain interaction network and the flow of the cryptocurrencies specifies the financial interactions, in Ethereum, *accounts* are associated with unique public addresses and transactions that take place among accounts constitute the interactions of the network [65]. Moreover, a transaction of Bitcoin may have *change* in the case when a UTXO is larger than the desired amount to transfer. However, in Ethereum, the exact amount is transferred among the accounts, hence there is no change. Besides, to aggregate multiple UTXOs for payment, or send money to many recipients in a single transaction, a Bitcoin transaction can have multiple inputs or outputs. However, an Ethereum transaction takes place from one sender to one recipient only. There are also major differences between Bitcoin and Ethereum in terms of the block processing time, how miners are awarded, the monetary supply, and transactions costs that make the two blockchain networks significantly

distinguishable [9, 17, 65, 227].

An interesting approach to investigate the problem of detecting malicious entities on a blockchain-based cryptocurrency network is by utilizing the power of efficient machine learning methods to automate the detection process. The objective of this work is to facilitate automatic tracking, aggregation, and analyzing activity records of different entities. An efficient novel framework based on supervised machine learning methods is proposed to analyze the transactional records of different entities, learn the behavior of the malicious and genuine ones, and predict the authenticity of the unseen entities. Our main contributions are as follows: **(1)** we collect a set of Ethereum public addresses associated with illegitimate activities such as *phish*, *hack*, *heist*, and *scam* through extensively searching online resources and consider them as *malicious* addresses. We extend this set with addresses interacting with the malicious addresses; **(2)** a set of features is extracted based on the transactional behaviors of entities. The importance of the feature extraction process is examined by comparing the performance of the classification of the raw dataset with that of the processed dataset generated through our proposed framework; **(3)** we employ different re-sampling techniques for alleviating the dataset imbalance issue, and conduct extensive experiments to investigate the impact of different techniques; **(4)** the proposed framework trains and evaluates several different classes of learning methods including ensemble classification methods, and utilizes the best-trained model for detecting the malicious entities in the dataset. Our extensive evaluations demonstrate that the proposed framework achieves high performance in classification of the Ethereum entities with average F_1 score of 0.996 for the ensemble methods.

3.2 Detecting Malicious Ethereum Entities

An overview of our proposed framework is shown in Fig. 3.1. Each module is presented in detail in the following subsections: first, we describe the data collection and parsing which are followed by the feature extraction and dataset pre-processing. The dataset generation algorithm including data collection, feature extraction, and pre-processing are represented in Fig. 3.2. We complete this section by the proposed solutions to the detection of malicious Ethereum entities.

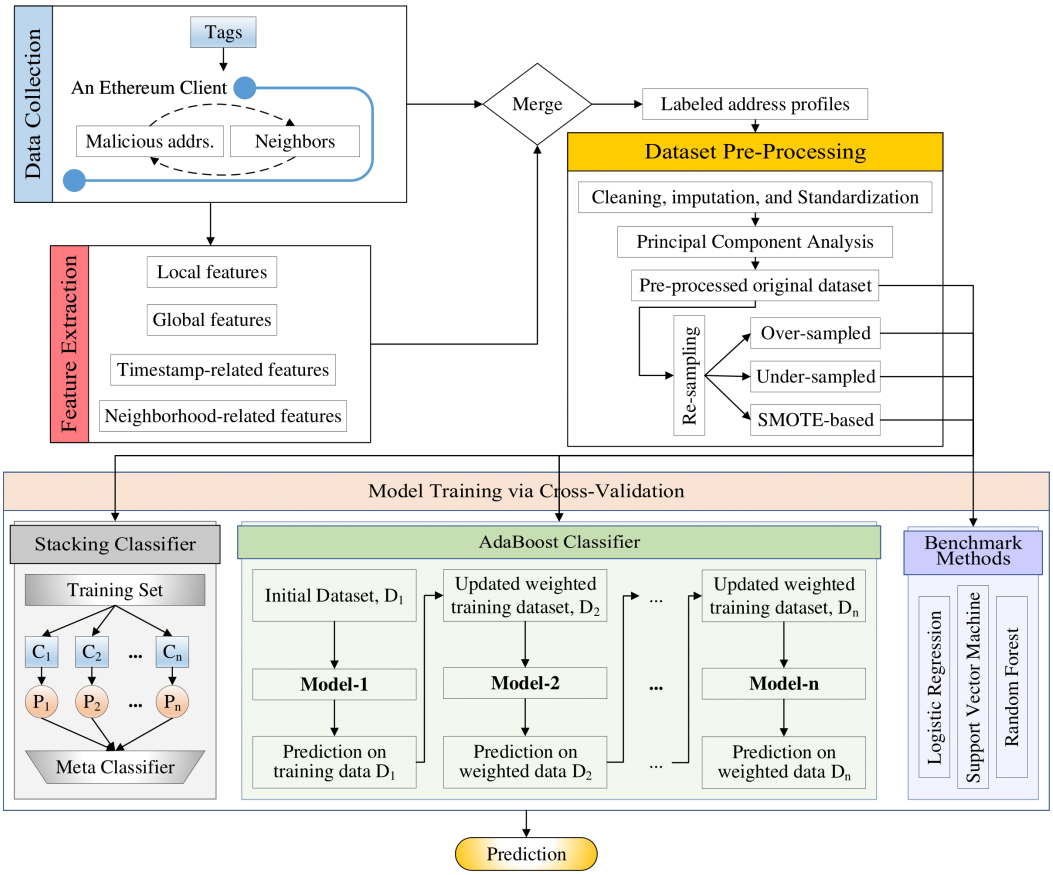


Figure 3.1: The proposed framework for detecting the Ethereum malicious entities.

Input:
lbl_list: target labels, specifically "*phish*, *hack*, *heist*, *scam*"
Output:
lbl_target_addr_list: the list of target addresses and their associated labels
addr_prof_ds: pre-processed dataset of the Ethereum address profiles

mal_addr_list = collect a list of addresses associated with the labels in *lbl_list*
for *addr* **in** *mal_addr_list*:
    neighbor_addr_list = find all the addresses interacting to *addr*
    target_addr_list += *neighbor_addr_list*
target_addr_list += *mal_addr_list*
for *addr* **in** *target_addr_list*:
    if *addr* **in** *mal_addr_list*:
        lbl[addr] = *"malicious"*
    else:
        lbl[addr] = *"genuine"*
    tx_hist[addr] = retrieve the tx history of *addr* from Ethereum ledger
lbl_target_addr_list = *target_addr_list* + *lbl*
feature_set = extract features from *tx_hist*
for *tx_set* **in** *tx_hist*:
    prof[tx_set] = generate the *feature_set values* for *tx_set*
addr_prof_ds = apply cleaning, imputation, standardization, and PCA to *prof*

Figure 3.2: A pseudocode presenting the dataset generation procedure.

3.2.1 Data Collection

One important step in our investigation is collecting and preparing the dataset for analysis. There are numerous entities on the Ethereum blockchain network with different transactional patterns, which make the behavioral pattern analysis non-trivial due to the complex diversity. These entities may belong to different types of addresses such as miners, exchanges, smart contracts, or advertising nodes. Each entity on the Ethereum network is uniquely identified by a public address. The public addresses are used for making transactions and transferring funds [11]. Besides, some meta-information about the addresses including the labels associated with some of the addresses can be retrieved by manually crawling the information available in the main Ethereum block explorer *etherscan.io*. These labels have been associated with the addresses by the Ethereum community as a result of demonstrating special behaviors. For instance, relevant labels are assigned to the addresses for showing special transactional patterns associated with illegal activities. In some other cases, several entities may become suspicious about the authenticity of an address, or an address may have a history of conducting illegal activities; thus, the malicious address is reported and a special label is assigned to that address.

It should be noted that there is no exhaustive database of labels for all the Ethereum entities, since the ever-increasing size of the Ethereum network data requires extensive investigation. Besides, such a comprehensive analysis is neither present nor fully plausible considering the dynamic nature of the Ethereum network. Therefore, only a very small portion of the addresses have special labels. On the contrary, it is assumed that the majority of the addresses on the network are honest with only a small number of entities involving in illegitimate activities. The labels associated with the known malicious addresses are retrieved from the Ethereum block explorer (i.e., *etherscan.io*), while the other addresses without any labels can be considered as being genuine, since they are not associated with illegitimate activities.

Aiming to find the addresses of entities associated with malicious activities, we searched through the main Ethereum block explorer *etherscan.io*, which enabled us to find 3128 addresses associated with *phish*, *hack*, *heist*, or *scam* activities. These addresses constituted our malicious address list. For analyzing the authenticity of different addresses on the Ethereum network, we aimed to generate a dataset consisting of behavioral profiles of different addresses. We start our dataset generation process by collecting a set of malicious addresses associated with illegitimate activities. Then, for

constructing our target list of Ethereum addresses, we find all different addresses that have sent Ether to or received Ether from any of the malicious addresses. The target address list includes a total of $53,087$ addresses where only $3,128$ of them are malicious ones and the rest of the addresses might belong to other categories of addresses such as exchange, miner, or they may not have specific labels. A pseudocode illustrating the overall dataset generation procedure is presented in Fig. 3.2. Specifically, at lines 1 to 11, the data collection processed is demonstrated. At line 1, a list of addresses associated with the target labels is collected. Then at line 2 to 5, the target address list consisting of the labeled addresses and all other addresses in interaction with them is constructed. Line 6 through 10 shows the assignment of labels to the target addresses. Finally, at line 11, the transaction history of each target address is retrieved from the Ethereum blockchain ledger. The feature extraction and dataset pre-processing are executed in lines 12 through 16.

For investigating the behavioral patterns of target addresses, we need to understand their transaction histories in the Ethereum public ledger. It is noteworthy that although the Ethereum blockchain ledger is publicly available, one of the most challenging tasks when investigating the blockchain data is gathering an appropriate large dataset for analysis. By joining the Ethereum network, we have access to the whole ledger, consisting of the total transactions history. However, processing the whole ledger requires huge processing power and memory requirements. Our proposed methodology does not require the investigation of all transactions on the ledger.

For retrieving the transaction histories of target addresses, we use the *Ethereum Development APIs* (from *etherscan.io*) to pull the transactions of each target address. Note that the Ethereum network is dynamic, so to feed recent information to the detectors, we adopt a *forget approach* [228, 229], taking the most recent $10K$ transactions of the target addresses into consideration (in case they have more than $10K$ transactions). In this way, we are able to pull the transaction histories of target addresses, consisting of more than $18M$ transactions (specifically, $18,686,447$). Having the transaction histories of target addresses, the next step extracts the appropriate features and prepares the dataset for classification.

3.2.2 Feature Extraction and Dataset Preparation

Using the transaction histories, this step extracts features that best describe the transactional behaviors of target addresses. The goal is to generate a dataset that can

be leveraged by the binary classifiers for detecting malicious entities. To examine the importance of extracting an efficient set of features and the dataset pre-processing, we investigated the classification on some of the initial raw attributes of the retrieved data from the Ethereum public ledger. Particularly, we started with considering four simple features describing the transactional behavior of each Ethereum entity: *(a)* the average value transferred, *(b)* the average amount of incentive for verification of all the transactions of that entity, *(c)* the average unit price of the incentive for all the transactions, and *(d)* the average amount of incentive that has been actually used for the transactions. We refer to the dataset consisting of these features as the *"raw"* dataset inferring that a limited simplistic set of initial attributes has been adopted. Moreover, no data pre-processing has been applied to the raw dataset. We applied three binary classification methods, *Logistic Regression (LR)* [230], *Support Vector Machine (SVM)* [231], and *Random Forest (RF)* [232], to detect the malicious entities.

Essentially, considering the necessity of the feature extraction process, our goal is to generate a set of features that best describe the characteristics of the Ethereum entities. Based on the different parameters of the transactions, e.g., the value sent/received and the incentive considered for the verification of the transactions, we extract a new set of features that could reasonably describe the overall behavior of entities. The extracted features form a behavioral profile for each entity that could be utilized by the classification methods for identifying the malicious entities. Each profile consists of 54 features:

- *General features*: including balance and active duration of the considered entity.
- *Neighborhood features*: including in-degree, out-degree, unique in-degree, and unique out-degree of the entity.
- *Local features*: including the aggregated value of parameters related to the incoming or outgoing transactions. These features include the minimum, maximum, average, standard deviation, and the total value of the transactions, the offered incentive for verification of the transactions, the incentive worth, and the actual amount of spent incentives for verifying the transactions.
- *Timestamp-related features*: including the minimum, maximum, average, and standard deviation of the time interval between incoming and outgoing transactions.

After constructing a behavioral profile for each entity, the aggregated dataset used for classification is constructed by augmenting the profiles of all target entities. As a

final step, we apply a data pre-processing and cleaning pipeline consisting of *imputation*, *standardization*, and *principal component analysis (PCA)* on our aggregated dataset in order to have clean and standardized features. The objective of PCA is to eliminate the probable correlation between the extracted features that can result in information redundancy. The principal components extracted by applying PCA are independent, so they do not have redundant information.

3.2.3 Tasks and Solutions

The challenge of detecting malicious entities in Ethereum is the accurate classification of a small number of malicious instances in a massive dataset. The goal is to reduce the false-positive rate without increasing the false-negative rate. Among the methods employed in anti-money laundering and fraud detection applications, RF and LR are two of the most widely used ones where the former is known for its high accuracy and the latter for its explainability [45, 56, 75, 77].

The task considered in this thesis is to monitor the transactional profiles of target Ethereum entities to assess their probabilities of conducting malicious activities. Specifically, in the final evaluation of the proposed detection process, each entity should be classified as malicious or genuine, and then by comparing the prediction results with the actual labels associated with the entities, the performance of the classification task is evaluated. The benchmark machine learning methods which are used in a supervised fashion for the binary classification task are LR, SVM, and RF.

We are tackling a classification problem where the legitimate entities far outnumber the malicious ones, thus the dataset is highly unbalanced. Most supervised learning methods are not designed to cope with highly imbalanced datasets, which makes it a difficult task [233]. For alleviating the imbalance problem of the dataset, we employ data-level re-sampling techniques including *under-sampling*, *over-sampling* [137], and *SMOTE (Synthetic Minority Over-sampling Technique)* [138] which were used as a pre-processing step to balance the dataset before the classification. When applying under-sampling, we down-size the genuine addresses (i.e. the majority class) by randomly omitting some instances until the dataset is balanced. This method reduces the data size and results in information loss. On the other hand, in our over-sampling process, we supplement the training dataset with multiple copies of some randomly chosen instances of the minority class (i.e. the malicious entities). Finally, when applying SMOTE, we augment the training dataset by synthetically generated instances of the minority class. Particularly,

we pick random elements of the minority class and compute their *k-nearest* neighbors. The synthetic instances are added among the selected elements and their neighbors. It should be noted that in different cases of re-sampling, we balance the dataset in such a way that the number of instances in both classes becomes almost equal.

As a result of applying different re-sampling techniques, we gain three additional versions of the dataset (i.e., under-sampled, over-sampled, and SMOTE-based dataset). For classifying the instances into malicious and genuine, we adopt two ensemble approaches for our binary classification problem namely *Stacking Classifier* and *AdaBoost* [231]. Essentially, an ensemble is a combination of models consisting of a series of other classifiers to create an improved classifier. For each instance in the dataset, the individual classifiers vote for a specific class, and the final prediction is returned based on a majority voting policy. Indeed, the ensemble learning methods are meta-algorithms which are a composite of several machine learning methods to increase the performance [231]. Specifically, stacking is a meta-learning approach in which there are two levels of classifiers. The first-level classifiers are learned on the original dataset. Then, a new dataset is constructed based on the prediction results of the first-level classifiers as new features. The second-level classifier is learned on the newly constructed dataset [231]. Alternatively, AdaBoost or *Adaptive Boosting* increases the accuracy of the classification by combining classifiers. AdaBoost is an iterative ensemble approach wherein each round, the probability or *weight* of selecting an instance of the training set is adjusted based on the learning performance of the previous round, and newly updated weight distributions are generated to be fed into the next round. In each round, higher weights are assigned to the instances which were incorrectly classified in the previous round. Therefore, the probability of selecting an incorrectly predicted instance for the next round is higher. This process ends either when a maximum number of iterations is reached or all the training data fits without any error. Our stacking classifier consists of a RF and a LR Classifier as the first-level classifiers and for the second-level classifier it employs another LR classifier. Besides, our version of AdaBoost uses RF as its base classifier. These two ensemble approaches together with our benchmark methods (i.e., LR, SVM, and RF) are evaluated on all four versions of the dataset (i.e., the original, under-sampled, over-sampled, SMOTE-based dataset) through a 10-fold cross-validation process. The best-trained model in each case is applied to an unseen set of data instances for evaluating the final prediction.

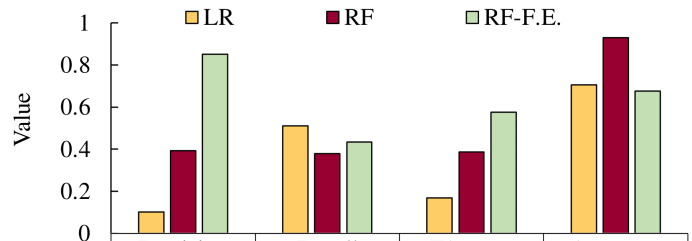
3.3 Experimental Evaluation

We applied our proposed malicious entity detection framework to the constructed Ethereum profiles dataset. We performed a 70:30 split of the training and test data, respectively. The majority of the entities on the blockchain network are honest, with only a small minority of the entities being malicious, so the considered dataset is highly imbalanced. It has been shown that for the unbalanced dataset *accuracy* is inadequate as a performance measure because if the classifier predicts all cases into the majority class, it will achieve a high accuracy value [75]. Therefore, although the classifier may show high accuracy, it does not indicate its high performance considering the dataset imbalance. Hence, we considered other performance indicators, such as *precision*, *recall*, and F_1 *score*, which take the imbalance issue into account.

3.3.1 Evaluating the importance of the feature extraction

The objective of this set of experiments is to investigate the importance of feature extraction and dataset pre-processing. Hence, the performance of the classification task for the raw dataset (presented in Section 3.2.2) is compared to that of the pre-processed dataset of the extracted features. In these experiments the proposed framework was employed for three standard classification methods namely LR, SVM, and RF (all from the *scikit-learn* [234] Python package with the following settings: LR with default parameters, SVM with *linear* kernel, and RF with 50 estimators and the maximum depth equal to 100). The results are illustrated in Fig. 3.3.

It should be mentioned that the SVM classifier could not converge in a reasonable time (specifically, even after 54 hours). As shown in Fig. 3.3, RF outperforms LR considering the raw data. However for the raw dataset, none of the methods demonstrate acceptable performance in terms of precision, recall, or F_1 score. These observations imply that the features considered in the raw dataset are not efficient in demonstrating the behavioral patterns of the entities. Next, to better realize the importance of the feature extraction and pre-processing, we examined the performance of the RF classifier when it is applied to the processed dataset (i.e., after applying the proposed steps to extract features and pre-process the data). As shown in Fig. 3.3, when the RF classifier is applied to the pre-processed dataset of extracted features (noted as *RF-F.E.*) considerably higher performance has been achieved. Therefore, extracting a set of features that can better describe the characteristics of the entities is



	Precision	Recall	F1 score	Accuracy
LR	0.101	0.51	0.169	0.706
RF	0.392	0.38	0.386	0.929
RF-F.E.	0.852	0.434	0.575	0.676

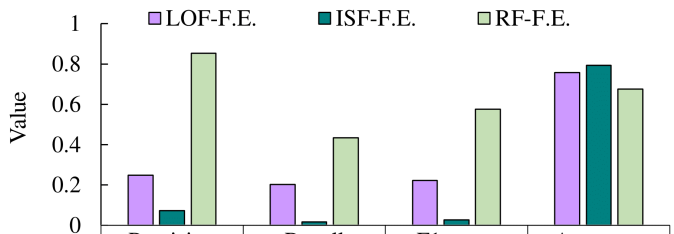
Figure 3.3: Assessing the importance of the feature extraction and dataset pre-processing of the proposed framework. The performance of LR and RF when classifying the raw dataset are compared to the RF when classifying the processed dataset of extracted features (noted as *RF-F.E.*).

imperative.

3.3.2 Evaluating the performance of the unsupervised vs. supervised classification methods

We also investigated the performance of unsupervised anomaly detection methods, namely *Local Outlier Factor (LOF)* [235] and *Isolation Forest (ISF)* [236], for detection of malicious Ethereum entities. LOF [235] is an unsupervised method which computes the local density of each point as its estimated distance to its k-nearest neighbors, and compares the local density of points to detect the anomalous points. It is important to note that LOF was adopted in [226] for detection of anomalous Bitcoin users and transactions. As another unsupervised method, we employed ISF which has shown good performance for high dimensional data [236].

We implemented LOF and ISF using the *scikit-learn* Python package [237] with default parameters and compared their performance with RF when all methods are applied to the pre-processed dataset of extracted features. The results are demonstrated in Fig. 3.4. As can be observed, LOF and ISF are not efficient in detecting the malicious Ethereum entities. The reason for the inefficiency of these anomaly detection methods is that malicious entities mainly endeavor to show behavioral patterns similar to normal entities in order to remain undetected. Thus, their malicious behaviors are not obviously different from the normal behavior of other entities, which make the



	Precision	Recall	F1 score	Accuracy
LOF-F.E.	0.2479	0.2024	0.2229	0.7563
ISF-F.E.	0.0735	0.0169	0.0275	0.7934
RF-F.E.	0.852	0.434	0.575	0.676

Figure 3.4: Comparing the performance of unsupervised (i.e. *LOF* and *ISF*) and supervised (i.e. *RF*) learning methods for classification of the pre-processed dataset of extracted features.

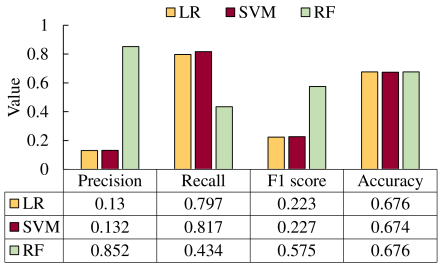
detection of malicious entities more challenging. Considering the inferiority of the unsupervised methods, we preferred to consider the supervised methods for detection of malicious entities in our framework.

Table 3.1: The Ethereum dataset specification.

Dataset		Malicious	Genuine	Total
all data		3,128	49,959	53,087
training set	original	2,190	34,971	37,161
	over-sampled	34,971	34,971	69,942
	under-sampled	2,190	2,190	4,380
	SMOTE-based	34,971	34,971	69,942
test set		938	14,988	15,926

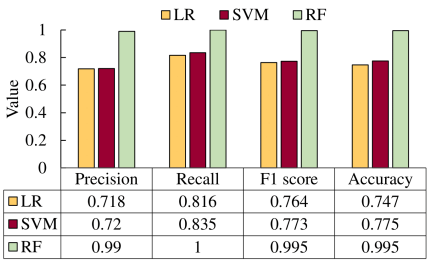
3.3.3 Evaluating the impact of dataset imbalance and different re-sampling techniques

In this set of experiments, we investigated the impact of the data imbalance on the performance of different classification methods such as LR, SVM, and RF. The considered dataset consisted of extracted features, and the re-sampling techniques were implemented to alleviate the imbalance issue. Each classification approach was evaluated on four distinct datasets (including the original, over-sampled, under-sampled, and SMOTE-based dataset) through a 10-fold cross-validation process. We generated SMOTE-based



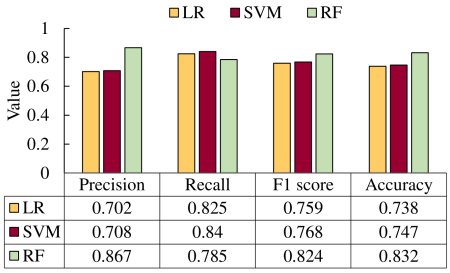
	Precision	Recall	F1 score	Accuracy
LR	0.13	0.797	0.223	0.676
SVM	0.132	0.817	0.227	0.674
RF	0.852	0.434	0.575	0.676

(a) Original dataset.



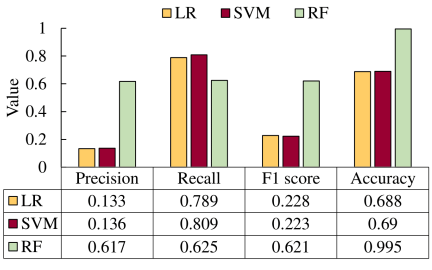
	Precision	Recall	F1 score	Accuracy
LR	0.718	0.816	0.764	0.747
SVM	0.72	0.835	0.773	0.775
RF	0.99	1	0.995	0.995

(b) Over-sampled dataset.



	Precision	Recall	F1 score	Accuracy
LR	0.702	0.825	0.759	0.738
SVM	0.708	0.84	0.768	0.747
RF	0.867	0.785	0.824	0.832

(c) Under-sampled dataset.



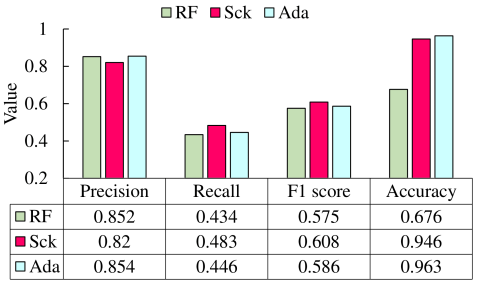
	Precision	Recall	F1 score	Accuracy
LR	0.133	0.789	0.228	0.688
SVM	0.136	0.809	0.223	0.69
RF	0.617	0.625	0.621	0.995

(d) SMOTE-based dataset.

Figure 3.5: Results of the experiments investigating the impact of dataset imbalance and the re-sampling techniques.

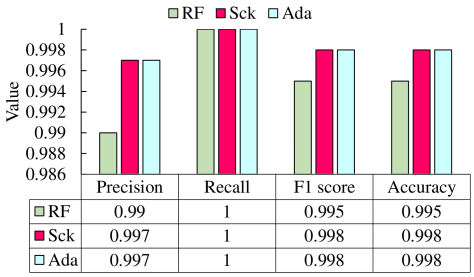
version of the dataset using the *imbalance-learn* Python API [238], and over- and under-sampled dataset using *scikit-learn* [234] Python package. The detailed specification of each different version of the dataset is presented in Table 3.1, and the performance results are summarized in Fig. 3.5. We observed that most learning classification methods do not perform well when the dataset is unbalanced. Moreover, detecting a positive instance (i.e., a malicious entity) and negative instance (i.e., a genuine entity) do not have a similar impact. False-negative cases (where a malicious entity has been incorrectly detected as genuine) have a more severe impact, since these cases may incur more further cost as a results of conducting illegitimate activities. Considering the results in Fig. 3.5, it can be observed that although most approaches show acceptable performance with regard to the accuracy, their precision, recall, and F_1 score are considerably different, with the original dataset having the lowest performance in terms of F_1 score in roughly all cases.

As can be observed in Fig. 3.5, among the methods of re-sampling, the best results were obtained with over-sampling in comparison with under-sampling and SMOTE. These findings can be explained by the fact that in the case of under-sampling, by randomly omitting some negative class instances from the original dataset to achieve the under-



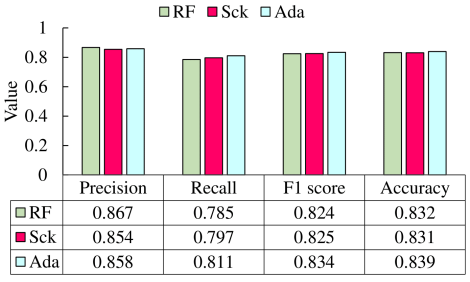
	Precision	Recall	F1 score	Accuracy
RF	0.852	0.434	0.575	0.676
Sck	0.82	0.483	0.608	0.946
Ada	0.854	0.446	0.586	0.963

(a) Original dataset.



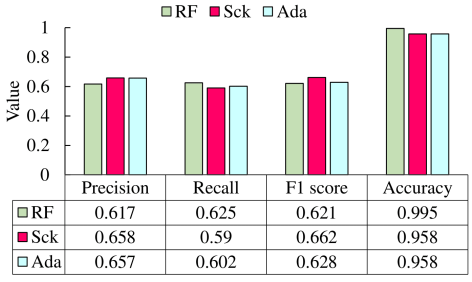
	Precision	Recall	F1 score	Accuracy
RF	0.99	1	0.995	0.995
Sck	0.997	1	0.998	0.998
Ada	0.997	1	0.998	0.998

(b) Over-sampled dataset.



	Precision	Recall	F1 score	Accuracy
RF	0.867	0.785	0.824	0.832
Sck	0.854	0.797	0.825	0.831
Ada	0.858	0.811	0.834	0.839

(c) Under-sampled dataset.



	Precision	Recall	F1 score	Accuracy
RF	0.617	0.625	0.621	0.995
Sck	0.658	0.59	0.662	0.958
Ada	0.657	0.602	0.628	0.958

(d) SMOTE-based dataset.

Figure 3.6: Evaluating the performance of different ensemble learning methods for the malicious entity detection.

sampled dataset, we lost a considerable portion of the information. Thus, the ability of the classifier to learn the dataset characteristics decreased, which resulted in lower performance. In the case of the SMOTE-based re-sampled dataset, the size of the positive class is increased by generating random instances that are in the neighborhood of the original positive instances. However, there are two important notes to consider. First, the number of positive instances are considerably lower than the negative instances. Second, the positive instances are malicious entities with different unique characteristics where there is no guarantee that they would show similarities in their behavioral patterns. Therefore, it is not guaranteed to find a distinguished community of malicious entities in all cases, rather each malicious entity being surrounded by genuine entities. With these notions in mind, the SMOTE-based dataset could not show very similar behaviors to the actual dataset, resulting in inferior performance for SMOTE-based dataset compared to the over-sampled dataset.

3.3.4 Evaluating the performance of the ensemble methods

Observing the superior performance of RF in comparison to the traditional classification methods, we extended our evaluations by exploiting two other ensemble classification methods for the detection of malicious entities. Particularly, we proposed a stacking classifier consisting of a RF and a LR as the first-level classifiers, and a LR as the second-level classifier. In addition, we utilized an AdaBoost approach consisting of a RF as its base classifier. The stacking classifier as well as the AdaBoost classifier were also applied to the four versions of the dataset. The results of these experiments are demonstrated in Fig. 3.6.

In general, ensemble methods have been developed to improve the classifiers' ability of generalization by combining information from multiple sources (such as prediction results of several classifiers) [231]. It can be observed that ensemble classification methods show high performance, while there are no significant differences in the performance of RF, Sck, or Ada in terms of F_1 score. In fact, the high performance of Sck and Ada mainly comes from the adoption of RF which has been proven to show superior performance in unbalanced classifications with usecases in fraud and anomaly detection [28, 45, 56, 75, 77]. Thus, other ensemble methods only present a slightly better performance than RF.

3.4 Conclusion

We proposed a novel framework for detecting malicious entities on the Ethereum network which incorporates data collection, feature extraction, and model training/evaluation with the goal of detecting malicious entities from public parameters of Ethereum transactions. By extensive evaluations, we deducted that the malicious behaviors on the Ethereum network may be detectable considering the transaction histories of the Ethereum entities. In addition, the proposed framework is easy to employ and it is adaptable to similar tasks. The evaluations demonstrated the good performance of the proposed framework for the ensemble learning methods including Random Forest, Stacking Classifier, and AdaBoost. Moreover, it is revealed that the extracted features were efficient in finding malicious entities on the Ethereum network.

Chapter 4

Graph-based Detection of Illicit Entities in Transaction Networks

While the proposed method in Chapter 3 is efficient in detecting malicious Ethereum addresses, we aim to employ the structure of the transaction networks, since relations of networks offer valuable information. In addition, employing relational attributes of the networks and exploiting graph-based analysis help in proposing richer feature sets. We are also motivated to have a unified generic pipeline for detecting illicit entities on cryptocurrency networks that encompasses both UTXO (e.g., Bitcoin) and account-based (e.g., Ethereum) architectural models. Hence, we move towards a graph-based approach.

4.1 Introduction

Blockchain-based cryptocurrencies, such as Bitcoin and Ethereum, have taken a considerable share of the financial market [9]. Malicious users increasingly exploit these platforms to undermine legal control or conduct illicit activities [9, 65, 239, 240]. In particular, billions of dollars attained through illegal activities such as drug smuggling and human trafficking are laundered smoothly through blockchain-based cryptocurrencies exploiting their pseudonymity [79]. Given the openness and transparency of blockchain [241], it is of paramount importance to mine this data for detecting such illicit activities.

Although the recent machine learning advances have enhanced the exploration of large-scale complex networks, the performance of these methods relies on the quality of the data representations and extracted features [242]. Likewise, in blockchain networks

with high anonymity and large number of participants with diverse transactional patterns, any illicit node detection method is effective only when the extracted characteristics of the data efficiently distinguish the illicit and licit components of the network. Hence, developing effective illicit node detection method depends heavily on the efficiency of the data representations and extracted features. Considering that network topologies can reflect the roles of the different nodes, graph representation learning methods have been potentially conceived as great means for capturing neighborhood similarities and community detection [149]. Additionally, machine learning analysis on large networks is becoming viable due to the efficiency, scalability and ease of use of graph representation learning methods [148, 149].

Driven by the need to enhance the security of the blockchain through transaction network analysis, and by recent advances in graph representation learning, we propose an efficient graph-based method SigTran for detecting illicit nodes in the transaction network of blockchain-based cryptocurrencies. SigTran can be applied for warning honest parties against transferring assets to illicit nodes. In addition to providing a trading ledger for cryptocurrencies, blockchain can be perceived as a network that analyzing its dynamic properties enhances our understanding of the interactions within the network [227, 243]. SigTran first constructs a graph based on the extracted transactions from the blockchain ledger considering integral characteristics of the blockchain network. Then it extracts structural, transactional and higher-order features for graph nodes; these features strengthen the ability to classify the illicit nodes. SigTran shows superior performance compared to the previous platform-dependant state-of-the-arts (SOTAs), while it is generic and applicable to different blockchain network models. Particularly, SigTran achieves an F_1 score of 0.92 and 0.94 for detecting illicit nodes in Bitcoin and Ethereum network, respectively. Moreover, SigTran is scalable and simpler compared to much more complex SOTA contenders. In short, SigTran is:

- **Generic:** SigTran is platform independent and applicable to different blockchain networks, unlike current contenders.
- **Accurate:** SigTran outperforms much more complex SOTA methods on the platforms they are designed for.
- **Reproducible**: we use publicly available datasets, and the code for our method and scripts to reproduce the results is available at: https://github.com/fpour/SigTran.

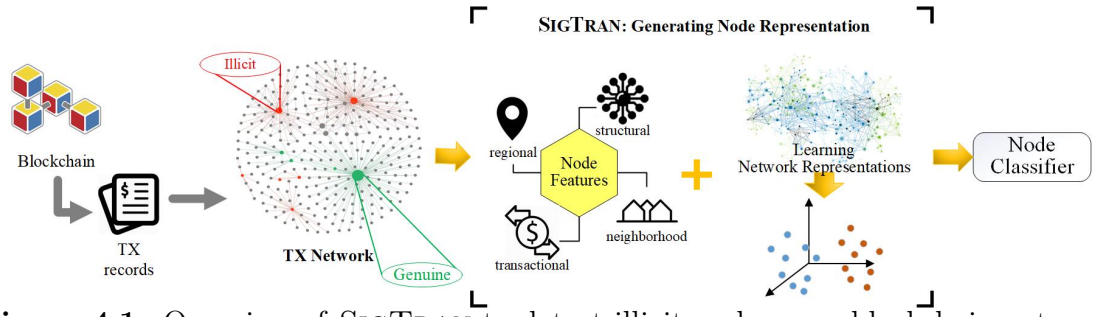


Figure 4.1: Overview of SIGTRAN to detect illicit nodes on a blockchain network.

4.2 Problem Formulation and Proposed Method

Given the publicity of the transaction ledger, our aim is to detect illicit activities on a blockchain-based cryptocurrency network. We formulate the problem as a node classification task in the transaction graph. Specifically, given the transaction records of a set of blockchain nodes, we devise the transaction graph and investigate the authenticity of different nodes by predicting the probability of each being involved in an illegitimate activities such as phishing, scam, malware, etc. We propose an efficient feature vector generation approach for nodes in these networks which demonstrates node activity signatures which can be used to distinguish illicit nodes. An overview of SIGTRAN framework is illustrated in Fig. 4.1. SIGTRAN extracts the transaction (TX) history from the blockchain ledger and constructs a transaction network from those records. To generate node representations, it then extracts a set of useful features which are fused with the corresponding node representations produced by a node embedding method. The final representations are then classified to detect illicit nodes. These steps are explained in detail in the following.

4.2.1 Transaction History Retrieval

The required transaction records can be obtained directly from the blockchain public ledger of the target cryptocurrency. For instance, for Bitcoin or Ethereum, we can use the client software of these peer-to-peer networks to pull down the blockchain data in binary format which is converted to human-readable formats like CSV via an appropriate parser. As an example, for converting the binary records of the Bitcoin and Ethereum ledger, *SoChain* [244] and [245] can be employed respectively. The transaction records contain details such as timestamp, amount sent or received, incoming and outgoing addresses, and other related information. Different authoritative websites (such as *EtherScamDB*

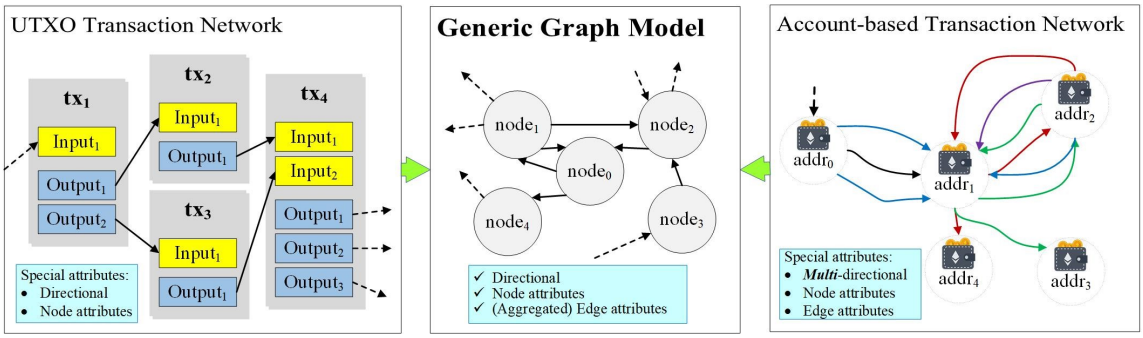


Figure 4.2: SigTran creates a generic graph model based on the transaction networks.

[246] for Ethereum network) helps in gathering a list of illicit nodes on the blockchain network. Together transaction records and the information about the authenticity of the network nodes constitute the dataset required.

4.2.2 Network Construction

A cryptocurrency transaction network is modeled as a graph demonstrating the interactions among participants of the network. We model a blockhchain network as a graph $G = (V, E)$, where V represents the set of nodes and E expresses the set of edges. Nodes and edges could have extra attributes, such as labels for nodes, and amount and timestamp of transaction for edges. Essentially, blockchain networks can be classified into two categories: (a) unspent transaction output *(UTXO)* model where the nodes specify the transactions, and the edges denote the flow of the cryptocurrency among nodes. Bitcoin, Dash, Z-Cash, and Litecoin are cyrptocurrencies based on the UTXO model [243], and (b) *account-based* model, where the account addresses are considered as the nodes and the transactions among addresses as the edges of the graph. Ethereum network is based on the account-based model. Considering the different categories of blockchain networks, we construct a generic graph model, as illustrated in Fig. 4.2, to which the instances of both the UTXO as well as the account-based network models are easily convertible. In the generic graph, the nodes specify the network entities in which we are interested to investigate their authenticity, while the edges denote the interactions among the nodes. The generated graph model entails any features associated with the nodes, whereas multiple edges between any two nodes with the same direction are aggregated into a single edge. It is noteworthy that based on the underlying blockchain network (i.e. UTXO or account-based), nodes and edges of the

generic graph can have different intuitions. Particularly, if the graph is constructed based on an UTXO blockchain, the nodes represent cryprocurrency transactions which may belong to licit or illicit categories of real entities. However, if the graph is constructed based on an account-based blockchain, each node represents either an illicit or licit address. In both cases, node representations and classification are applied incognizant of the underlying blockchain model.

4.2.3 SigTran

SigTran*: Signature Vectors for Detecting Illicit Activities in Blockchain Transaction Networks*

After modeling the blockchain transactions network as a graph, we need to develop proper representations for the nodes. This consists of a set of carefully crafted features which are fused with learned node representations, explained below respectively.

SigTran-Feature Extraction

For each node u, we gain a diverse set of features consisting of four main categories as follows. It is important to note that the features of the nodes (e.g., labels) and edges (e.g., amount and timestamp) of the original network are preserved in the constructed generic model, since we employ these attributes for extracting the features of the nodes.

- *Structural features* consist of in-degree ($D_{in}(u) = \sum_{v \in N_u} |e_{vu}|$), out-degree ($D_{out}(u) = \sum_{v \in N_u} |e_{uv}|$), and total degree ($D_{tot}(u) = D_{in}(u) + D_{out}(u)$) of node u. As there may exist multiple edges between two nodes, $|e_{vu}|$ determines the number of edges from v to u, and N_u consists of all first-order neighbors of node u.
- *Transactional features* investigate the characteristics related to the amount and time interval of the transactions. Indeed, blockchain specific information of the transaction network is mainly enriched in this set of features. Each edge e_{uv} from u to v is associated with a set of attributes including the amount and time interval of the transactions from node u to node v. For obtaining transactional features, we consider a set of aggregation functions, G, which includes *summation*, *maximum*, *minimum*, *average*, *standard deviation*, and *entropy* operations over an arbitrary given set of values X as follows:

$$G = \{\sum(\{X\}), \max(\{X\}), \min(\{X\}), \overline{\{X\}}, \sigma(\{X\}), H(\{X\})\} \tag{4.1}$$

With the set of aggregation functions G, transactional features of node u are defined as:

$$tx_u^{amnt} = \{g(\{e_u^a\}) \mid g \in G, e_u^a \subseteq \{e_{uv}^a, e_{vu}^a\}\}, tx_u^{freq} = \{g(\{e_u^\tau\}) \mid g \in G, e_u^\tau \subseteq \{e_{uv}^\tau, e_{vu}^\tau\}\}$$

where e_u^a denotes the amount related to (in/out) edges of node u. Similarly, e_u^τ denotes the time interval related to (in/out) edges of node u.

• *Regional features* are defined with regard to the ego network of a node. We consider the *egonet* of node u ($S_u = (V_u, E_u)$) as a subgraph of the original graph consisting of u and its first-order neighbors (i.e. N_u), with all the edges amongst these nodes. As an example, considering the generic graph model in Fig. 4.2, the egonet of $node_0$ consists of $\{node_1, node_2, node_4\}$. Having the definition of the egonet in mind, we consider the number of edges of S_u as one of the regional features of node u. Besides, the in-degree, out-degree, and total degree of S_u are considered as the other regional features according to $D_{in}(S_u) = |\{e_{wv} \in E \mid w \notin V_u, v \in V_u\}|$, $D_{out}(S_u) = |\{e_{wv} \in E \mid w \in V_u, v \notin V_u\}|$, and $D_{tot}(S_u) = D_{in}(S_u) + D_{out}(S_u)$, where $V_u = u \cup N_u$.

• *Neighborhood features* analyze the aggregated characteristics of neighbors of node u. Considering the aggregation functions in (5.1), the neighborhood features of node u are defined as: $D_{in}(N_u) = \{g(\{D_{in}(v)\}) \mid g \in G, v \in N_u\}$, $D_{out}(N_u) = \{g(\{D_{out}(v)\}) \mid g \in G, v \in N_u\}$, and $D_{tot}(N_u) = \{g(\{D_{tot}(v)\}) \mid g \in G, v \in N_u\}$.

Network Representation Learning

In order to learn node representations which fuse topological perspectives of the nodes in a cryptocurrency transaction network, SigTran combines the extracted features explained in above (which are obtained focusing on the specific characteristics of the cryptocurrency networks such as amount and time interval of transactions) with the node representations that are learned automatically through a network embedding procedure. For retrieving more efficient node representations, we exploit a common network embedding method for learning the features of the nodes in the generic graph model. Then, we fuse the extracted features with the node embeddings in an effective manner so that the ultimate node representations effectively demonstrate the fundamental characteristics of the nodes. For fusing the extracted features and the node embeddigns, we investigate two approaches explained in the following subsections.

RiWalk-enhanced. In this approach, we focus on the fact that nodes with different functionalities have different roles in a network, and the structure of the network can be investigated for gleaning these roles [149]. Hence, we consider the SigTran-features

as powerful indicators of similarity among nodes, and decouple the node embedding procedure into two steps. First, we identify the top ten SigTran-features with the highest importance in detecting the illicit nodes and retrieve the values of those features for each node u as $\mathbf{f}_u^*$. We then relabel each neighbor of node u such as v according to the function $\phi(v) = h(\mathbf{f}_u^*) \oplus h(\mathbf{f}_v^*) \oplus d_{uv}$. Here, d_{uv} denotes the shortest path length from u to v, $\oplus$ is the concatenation operation, and $h(x)$ is defined as $h(x) = \lfloor log_2(x+1) \rfloor$. The new labels which are generated based on the node features infer the role of the nodes (thus, *Ri: Role identification*). Thereafter, the second step consists of a random-walk-based network embedding method for learning the node representations. Specifically, we generate several random walks starting from each node, then merge the random walks to construct a corpus and adopt the Skip-Gram model with negative sampling of *word2vec* [150] to learn the node representations.

SigTran. In this approach, we consider the fusion of the SigTran-features and automatically generated node embeddings through a concatenation procedure. Particularly, we apply a random-walk-based node embedding method such as node2vec [148] and for each node u obtain its embedding as $\mathbf{e_u^*}$. Then, we generate the final representations by concatenating the SigTran-features $\mathbf{f_u^*}$ with the node embeddings for each node (i.e., $\mathbf{e_u^*} \oplus \mathbf{f_u^*}$) intending to achieve accurate node representations.

4.2.4 Node Classification

The generated node representations can then be used in the downstream task for classification of the illicit and genuine nodes. The illicit node detection task is akin to the common task of fraud detection and anti-money laundering applications. We simply employ *Logistic Regression* for the classification task because of its widespread adoption in similar tasks as well as its high interpretability [45, 56, 75, 77]. This simple choice enables us to better compare the effect of different embedding techniques.

4.3 Experimental Analysis

This section evaluates SigTran experimentally.

4.3.1 Datasets

We investigated two real-world transaction datasets consisting of the most widely adopted cryptocurrencies: (a) Bitcoin blockchain network which is the largest cryptocurrency

Table 4.1: Statistics of the investigated Blockchain-based Cryptocurrency Networks.

Dataset	Nodes	Edges	Illicit Nodes	Average Degree
Bitcoin	203,769	234,355	4,545	1.3002
Ethereum	2,973,489	13,551,303	1,165	9.1148

Table 4.2: SigTran outperforms baselines on the **Bitcoin** dataset. The average and standard deviation of 10 different runs are reported. The last three rows are introduced in this study.

Algorithm	**Precision**	**Recall**	F_1	**Accuracy**	**AU-ROC**
Weber et al. [45]	0.901 (±0.011)	0.929 (±0.008)	0.915 (±0.007)	0.913 (±0.008)	0.976 (±0.003)
node2vec	0.627 (±0.028)	0.312 (±0.031)	0.415 (±0.028)	0.563 (±0.013)	0.580 (±0.020)
RiWalk	0.549 (±0.016)	0.343 (±0.039)	0.421 (±0.030)	0.530 (±0.010)	0.547 (±0.014)
RiWalk-enhanced	0.582 (±0.041)	0.486 (±0.140)	0.522 (±0.100)	0.573 (±0.047)	0.619 (±0.077)
SigTran-Features	**0.905** (±0.008)	0.926 (±0.005)	0.915 (±0.004)	0.914 (±0.004)	0.976 (±0.002)
SigTran	0.890 (±0.010)	**0.947** (±0.008)	**0.918** (±0.006)	**0.915** (±0.006)	**0.976** (±0.003)

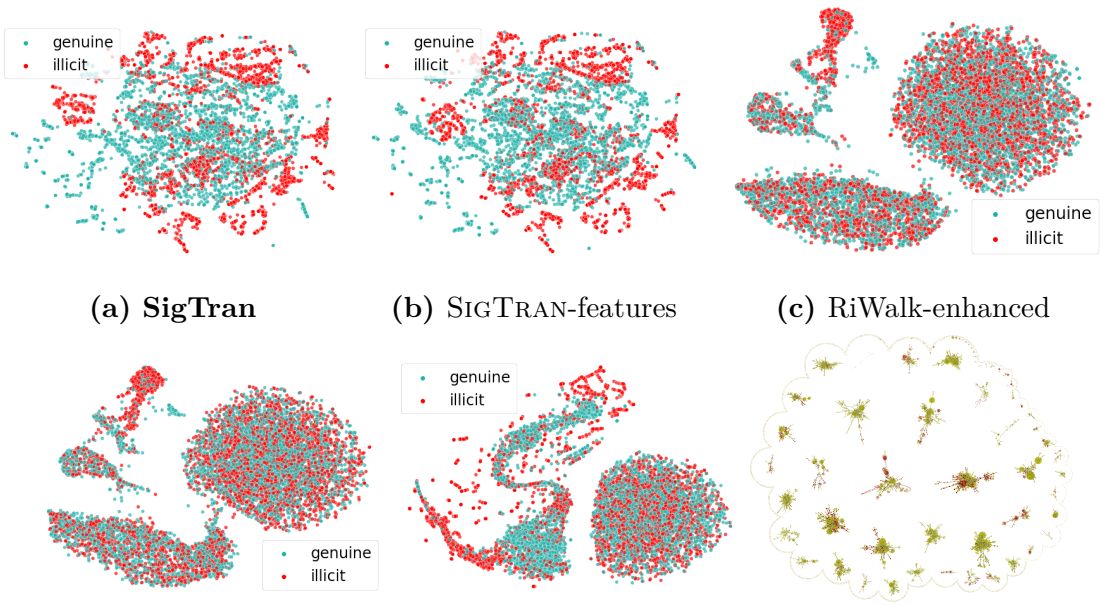
system based on UTXO model, and (b) Ethereum that support smart contracts, holds the second largest cryptocurrency, and provides an account-based model.

We employed Bitcoin transactions dataset shared by Weber et al. [45] in which 21% of the transactions are labeled as *licit* (corresponding to different legitimate categories such as exchanges, miners, wallet provider, etc.), 2% as *illicit* (corresponding to different categories of illegitimate activities such as scams, malware, ponzi scheme, ransomeware, etc.), and there are no labels for the rest of the transactions. In addition to the transaction records, the Bitcoin dataset consists of a set of handcrafted features representing the characteristics of the considered transactions. Since the dataset is fully anonymized, we could only generate *structural*, *regional*, and *neighborhood* features for the nodes of the Bitcoin graph. We combined SigTran-features with the initial attributes available in the dataset to form the node features of the Bitcoin network. In addition, we investigated the Ethereum transactions data shared by Wu et al. [85]. This dataset consists of Ethereum transaction records for a set of addresses consisting of licit addresses as well as illicit addresses reported to be involved in phishing and scam activities. The statistical details of the Bitcoin and Ethereum dataset are shown in Table 4.1.

4.3.2 Baseline Methods

Several SOTA methods were evaluated and compared.

- *node2vec* [148] is a random walk-based node representation method which employs



(a) SigTran **(b)** SigTran-features **(c)** RiWalk-enhanced

(d) RiWalk **(e)** node2vec **(f)** TX network

Figure 4.3: Bitcoin Embeddings: SigTran better separate illicit (red) and genuine (blue) transactions in Bitcoin network (plotted in (f)) compared to other baselines.

biased random walks to explore the neighborhood of the nodes with the consideration of local and global network similarities. Default parameters of the node2vec are set in line with the typical values mentioned in the paper [148]: context size $k = 10$, embedding size $d = 64$, walk length $l = 5$, and number of walk per node $r = 20$. We have also considered setting $p = 0.25$ and $q = 4$ to better exploit the structural equivalency of the nodes according to the discussion in the paper [148].

- *RiWalk* [149] is another random walk-based node embedding methods which focuses on learning structural node representations through decoupling the role identification and the network embedding procedures [149]. We considered the *RiWalk-WL* which aims to imitate the neighborhood aggregation notion of the Weisfeiler-Lehman graph kernels, and captures fine-grained connection similarity patterns.

We also compared the performance of SigTran with methods specifically designed for Bitcoin or Ethereum network.

- Bitcoin: we considered the method proposed by *Weber et al. [45]* as the baseline.

Table 4.3: SigTran outperforms baselines on the **Ethereum** dataset. The average and standard deviation of 10 different runs are reported. The last three rows are introduced in this study.

Algorithm	Precision	Recall	F_1	Accuracy	AU-ROC
trans2vec [85]	0.919 (±0.017)	0.894 (±0.019)	0.906 (±0.013)	0.908 (±0.012)	0.967 (±0.006)
node2vec	0.917 (±0.016)	0.907 (±0.019)	0.912 (±0.015)	0.912 (±0.015)	0.964 (±0.007)
RiWalk	0.931 (±0.016)	0.764 (±0.027)	0.838 (±0.017)	0.853 (±0.015)	0.894 (±0.011)
RiWalk-enhanced	0.928 (±0.017)	0.832 (±0.038)	0.877 (±0.022)	0.884 (±0.019)	0.899 (±0.015)
SigTran-Features	0.923 (±0.016)	0.926 (±0.005)	0.925 (±0.008)	0.925 (±0.009)	0.958 (±0.006)
SigTran	**0.944** (±0.014)	**0.940** (±0.012)	**0.942** (±0.008)	**0.942** (±0.008)	**0.976** (±0.005)

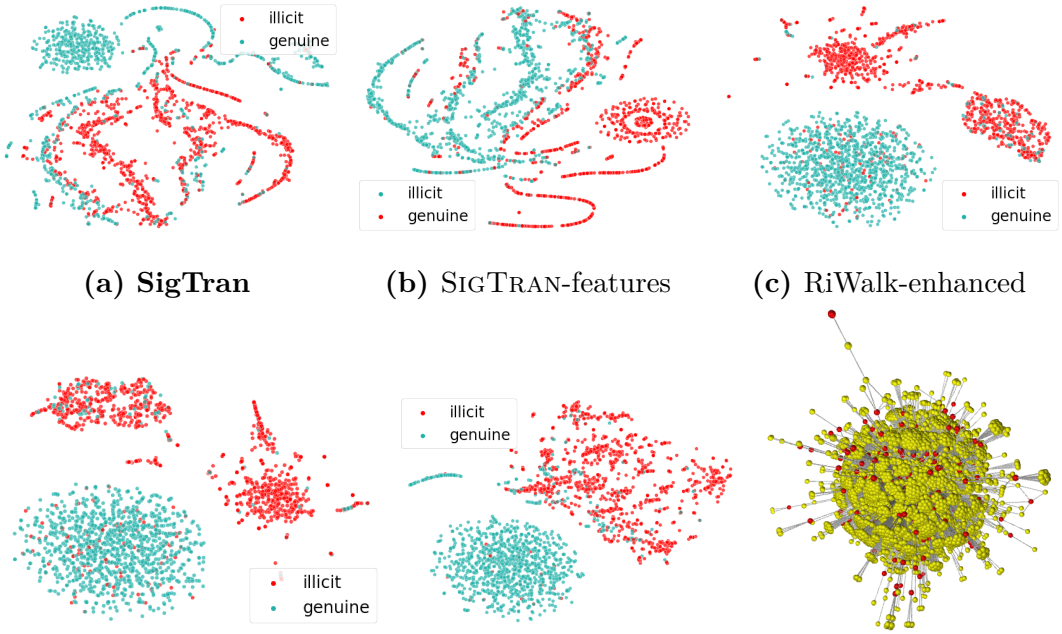
- Ethereum: we considered phishing scams detection method by Wu et al. [85] denoted as trans2vec as the baseline method. To make a fair comparison, we set the default parameters of trans2vec inline with the parameters of the node2vec.

4.3.3 Performance Evaluation

To evaluate the performance of SigTran, we considered the illicit nodes as the target of the detection approach and randomly selected an equal number of genuine nodes to form our set of anchor nodes. We extracted the first-order neighbors of all the anchor nodes and all edges among these nodes to construct a subgraph for investigation. Random selection of genuine nodes was repeated for 50 times, thus 50 different subgraphs were examined and the average performance was reported. Logistic regression with *L1* regularization was implemented in Scikit-learn Python package as the node classifier. The performance evaluation results for the Bitcoin and Ethereum network are illustrated in Table 4.2 and Table 4.3, respectively. To investigate the importance of *SigTran-features*, both tables also report the performance of the classification tasks when only SigTran-features were used as the node representations.

Bitcoin

Considering the results of illicit node detection on Bitcoin network in Table 4.2, it can be observed that node embedding methods namely node2vec and RiWalk did not generate efficient node representations. Therefore, the classification task had very low performance in detecting illicit nodes. The poor performance of node2vec and RiWalk is due to the fact that these methods are not specifically dealing with the intrinsic characteristics of financial networks, such as having multiple edges among nodes, or



(a) SigTran **(b)** SigTran-features **(c)** RiWalk-enhanced

(d) RiWalk **(e)** node2vec **(f)** TX network

Figure 4.4: Ethereum Embeddings: SigTran better separate illicit (red) and genuine (blue) accounts in Ethereum network (plotted in (f)) compared to other baselines. Notice the red nodes mixed in the blue cluster in (c-e).

being dependent on the amount and time interval of the transactions. These methods mainly focus on exploiting the structural similarities in order to maximize the likelihood of preserving neighborhoods of nodes. However, the results demonstrate that ignoring the specific characteristics of cryptocurrency networks, such as amount and timestamp of the transactions, results in embeddings that are not efficient for achieving decent illicit node classification performance. On the other hand, methods likes Weber et al. [45] and SigTran that are designed specifically for cryptocurreny networks show much better performance. Superior performance of SigTran compared to Weber et al. [45] is due to its extended set of features as well as the exploitation of the structural information via node embedding methods. It is noteworthy to mention that SigTran is more efficient than the proposed RiWalk-enhanced method. This can be attributed to two main reasons. First, in RiWalk-enhanced, we employed the extracted features only for relabeling the nodes. Although the labels of the nodes impact the node embeddings, the exact values of the extracted features do not directly influence the embeddings values which are later used for the node classification task. Moreover, it should be

noted that the new labels combine the extracted features of the anchor and neighbor nodes as well as their shortest path distance. Thus, modified values of the extracted features are used for labeling. However, it is noteworthy that RiWalk-enhanced outperforms its counterpart RiWalk, which underlines the importance of fusing the extracted features with the node embeddings in terms of improving the performance of the node classification task. For a qualitative comparison of the different embedding methods, we have depicted the t-SNE [247] transformations of different node representations methods for one of the subgraphs of the Bitcoin network in Fig. 4.3. According to Fig. 4.3, it can be observed that the embeddings produced by SigTran shape more separable distributions.

Ethereum

For the Ethereum dataset as shown in Table 4.3, it can be observed that SigTran demonstrates considerably better performance than the other methods. Although trans2vec and node2vec demonstrate high performance, the superior performance of SigTran underlines its efficiency in employing the native characteristics of the cryptocurrency networks as well as structural information obtained by the node embedding methods. Besides, we can observe that the extracted features improved the performance of the RiWalk-enhanced compared to RiWalk. Due to the fact that SigTran better incorporates the extracted features with the network structural embeddings, it achieves the most decent performance on the Ethereum network as well. We have also depicted t-SNE [247] transformations of different node embedding methods for a subgraph of the Ethereum network in Fig. 4.4. Considering Fig. 4.4, it is observable that embeddings obtained by SigTran show considerable distinction between illicit and licit nodes, while for example in Fig. 4.4e, there are several illicit nodes (marked with red) in the licit cluster (marked with blue).

4.4 Conclusion

We propose SigTran that extracts signature vectors for detecting illicit activities in blockchain network. Our proposed SigTran transforms the blockchain network into a simple graph and then extracts carefully designed features which explain structural, transactional, regional, and neighborhood features of the nodes. These features are then combined with generic node representations which encode the roles of the nodes in a given

graph. SigTran should be considered as a simple and strong baseline when developing more complex models. Our proposed SigTran baseline is:

- **Accurate**: SigTran outperforms state-of-the-art alternatives in detecting illicit activities in blockchain transaction records.
- **Generic**: SigTran is platform independent and we apply it to blockchain data extracted from both Bitcoin and Ethereum.

Chapter 5

Graph-based Anomaly Detection in Temporal Graphs

Considering the performance of our proposed approaches for detecting malicious entities in cryptocurrency networks in Chapter 3 and Chapter 4, in this Chapter, we discuss how the research problem and the proposed approaches can be extended to the node classification task in temporal graphs.

5.1 Introduction

Various real-world complex systems can be abstracted by temporal networks that describe relations or interactions. Temporal networks have ubiquitous applicability in wide range of domains such as cryptocurrency transactions networks [19], social networks [221], and recommendation systems [218, 220]. Recently, extensive research has been conducted over graph structured data to learn vector representations of network elements such as nodes or interactions [153, 156, 219, 220, 221, 222]. Although temporal evolution is a principal property of many applications, research has mainly focused on static graphs where network elements or their attributes are fixed over time [142]. In representation learning on temporal graphs, it is important to note that topological structure as well as node and edge features are evolving over time [220]. Therefore, it is necessary to efficiently integrate temporal, structural, and semantic characteristics of the networks elements.

A fundamental task in network analysis is node classification where the objective is to categorize the node into one of the predefined classes [142, 147]. Node classification

has many practical applications on real-world networks. In particular, it can be adopted to predict the authenticity of network users to prevent illicit activities. In fact, as a side effect of the relentless growth of various networks, increasing opportunities exist for the malicious parties to take advantage of manipulating the network data in different context: from cryptocurrency transaction networks (e.g. scammers), to social network (e.g. trolls), to e-commerce interaction networks (e.g. fraudsters) and many more [248]. For instance, in e-commerce networks where consumers make decisions based on the user-generated contents, such as ratings and reviews, there is a financial motivation to manipulate the network by fake ratings [120, 249]. Similarly, rating platforms also exist in the context of cryptocurrency transaction networks, where participants examine reviews and ratings of an online service or product as a measure of trust to decide about whom to trade cryptocurrency with. Thus, fraudulent users find significant incentives to give fake ratings [250].

Considering the huge economical and social impacts of conducting malicious activities on large networks, considerable attention has been given to protecting the networks [248]. In particular, a large body of data mining and machine learning techniques have been developed for examining the interactions of the users within a network and discovering potentially malicious activities. Among them, the most effective ones are the graph-based approaches such as [45, 85, 120, 249], which look at the activity patterns summarized as a network structure. Here, we follow this graph-based line of work and propose a strong baseline for node classification in temporal graph that outperforms more expensive state-of-the-art contenders (see Table 5.1).

In this part of the thesis, we model online interaction networks, including cryptocurrency transaction networks, rating networks, and social networks, as weighted temporal graphs. In these commonplace graphs, there is a time associated with each interaction between users, as well as an intensity (or weight) and direction. Our proposed approach, *TGBase*, incorporates interaction-based, as well as structural and semantic attributes of the nodes to set out a carefully-designed set of features for each node with no learning needed or parameters to adjust. TGBase provides simple yet effective node representations that can be employed for distinguishing different types of nodes. We compare TGBase with many state-of-the-art models which learn representations from the data often using complex deep models. We show that our shallow model which incorporates these effective features outperforms these contenders while being a simpler and more efficient approach for node classification in *both* static and dynamic setting. The main contributions of this part of the thesis are three-fold:

Table 5.1: TGBase meets all desirable properties. *: Network embedding methods. †: Graph neural networks methods. ‡: Temporal graph representation learning methods.

	NE*		GNN†			TGRL‡				Platform-dependent			
	node2vec [148]	RiWalk [149]	GCN [154]	GraphSAGE [155]	GAT [156]	JODIE [218]	DyRep [219]	TGAT [220]	TGN [221]	Weber et al. [45]	trans2vec [85]	REV2 [120]	**TGBase**
Uses network structure	✔	✔	✔	✔	✔		✔	✔	✔		✔	✔	✔
Uses interaction information						✔	✔	✔	✔	✔	✔	✔	✔
Considers network dynamics						✔	✔	✔	✔		✔		✔
Parameter free										✔		✔	✔
Scalable				✔	✔	✔	✔	✔	✔				✔
Generic	✔	✔	✔	✔	✔	✔	✔	✔	✔				✔

- We propose an *efficient* method, TGBase, for node classification in temporal interaction graphs.
- We conduct extensive experiments, on large scale graphs with millions of edges, to show TGBase is *accurate*, *scalable* and *general*.
- We release TGBase as a simple baseline for node classification in temporal graphs. The TGBase-features are simple and easy to generate.

Reproducibility: Our code is open-sourced at `https://github.com/fpour/TGBase`. This includes the datasets and scripts for reproducing the reported results.

5.2 TGBase for Node Classification

We devise comprehensive node encodings to represent semantic, structural, and temporal attributes of each node in a temporal network useful to distinguish different types of nodes. The network is modeled as a weighted temporal graph $G = (V, E)$, where V and E denote the sets of nodes and edges, respectively. For each edge $e_{uv} \in E$ from node $u \in V$ to node $v \in V$, we denote its intensity by $w(e_{uv})$, and its timestamp by $t(e_{uv})$.

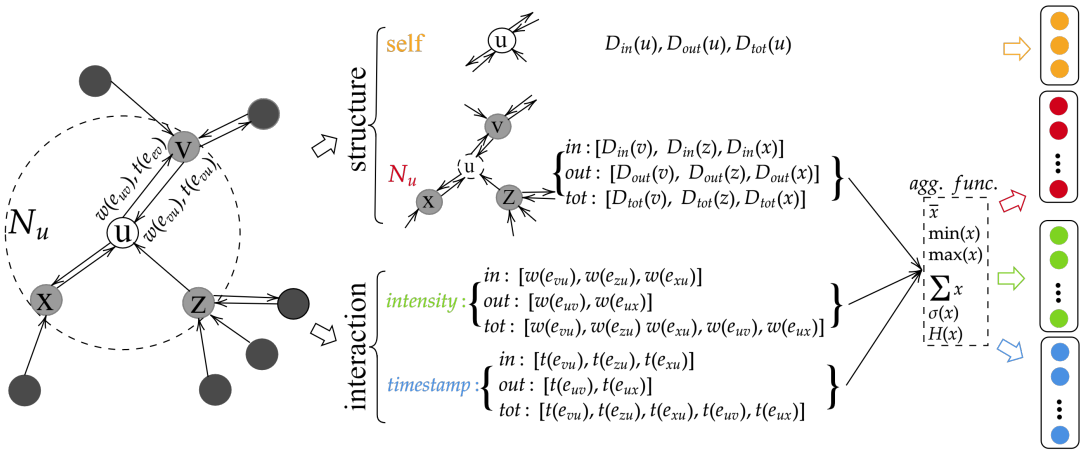


Figure 5.1: TGBase features for a given node u incorporates two types of local features: *structure*-based features (*self* and *neighborhood*), and *interaction*-based features (*intensity* and *timestamp*).

5.2.1 TGBase

For each node u in a given weighted temporal directed graph, we extract a set of features. The TGBase features include two main types (see Fig. 5.1): i), the ***structural*** attributes of node u (*self*) and its neighboring nodes (*neighborhood*); ii) the ***interaction***-*based* features that specify the *intensity* and *timestamp* of interactions between u and its neighbors. More specifically we have:

• *Structure*-***Self*** features, which consist of in-degree, out-degree, and total degree of u. More formally defined as: $D_{in}(u) = \sum_{v \in V} |e_{vu}|$, $D_{out}(u) = \sum_{v \in V} |e_{uv}|$, and $D_{tot}(u) = D_{in}(u) + D_{out}(u)$, respectively.

• *Structure*-***Neighborhood*** features, which aggregate the structural attributes of the neighbors of node u. We consider a set of six aggregation functions, denoted by A, consisting of *average*, *maximum*, *minimum*, *summation*, *standard deviation*, and *entropy* over an arbitrary set of values, denoted by X. More specifically:

$$A = \{\overline{\{X\}}, \max(\{X\}), \min(\{X\}), \sum(\{X\}), \sigma(\{X\}), H(\{X\})\} \tag{5.1}$$

Given the first-order neighbors of node u, $N_u = \{v \,|\, e_{uv} \in E \vee e_{vu} \in E\}$, we first derive the three structural attributes of each neighbour v, i.e. $[D_{in}(v),\ D_{out}(v),\ D_{tot}(v)]$, and then apply the six aggregation functions in Eq. (5.1) over each of these three attributes to get our 18 aggregated neighborhood features for u. Table 5.2 provides the complete

list of the resulted features.

• *Interaction-**Intensity*** features express the aggregated statistics of the intensity of interactions among node u and its neighbours. We consider three sets of edges, incoming, outgoing and total edges, and aggregate their intensity with the same aggregation functions in (5.1) to gain 18 edge-weight or intensity related features. More specifically: $W_{in}(u) = \{g(w_u^{in}) \mid g \in A, w_u^{in} = \{w(e_{vu}) \mid v \in N_u\}\}$, $W_{out}(u) = \{g(w_u^{out}) \mid g \in A, w_u^{out} = \{w(e_{uv}) \mid v \in N_u\}\}$, and $W_{tot}(u) = \{g(w_u^{tot}) \mid g \in A, w_u^{tot} = \{w(e_{vu}), w(e_{uv}) \mid v \in N_u\}\}$. This generalizes to multi-graphs, where there may exist several edges among each pair of nodes, since it is inherently summarizing the characteristics of multiple interactions.

• *Interaction-**Timestamp*** features express the aggregated statistics of interval of interactions among node pairs. Considering the interaction time interval of node u and its neighbor v together with aggregation functions in (5.1), the timestamp features of node u are acquired as follows: $T_{in}(u) = \{g(t_u^{in}) \mid g \in A, t_u^{in} = \{t(e_{vu}) \mid v \in N_u\}\}$, $T_{out}(u) = \{g(t_u^{out}) \mid g \in A, t_u^{out} = \{t(e_{uv}) \mid v \in N_u\}\}$, and $T_{tot}(u) = \{g(t_u^{tot}) \mid g \in A, t_u^{tot} = \{t(e_{vu}), t(e_{uv}) \mid v \in N_u\}\}$.

The aforementioned features are summarized in Table 5.2. In total, we defined only 57 local features per node. It is worth noting that for preserving semantic information of the nodes, we concatenate nodes initial attributes with the TGBase features in case the graphs are attributed, i.e. explicit features are provided for nodes.

5.2.2 Static vs. Dynamic Node Classification.

TGBase can provide node representations for *static* as well as *dynamic* node classification. Traditionally, node classification on graphs have been considered in a static setting, meaning that the categories of the nodes are fixed, and based on the observed interactions of the nodes, we predict the node classes. Therefore, it suffices to construct only one representation per node to be exploited by the downstream classification task. In this setting, we construct TGBase representations only once based on the observed history of the evolution of the graph.

However, the evolution of the temporal graphs can be considered as a sequence of timestamped events where at each timestamp, a new decision regarding the classes of the nodes should be made. This dynamic node classification setting implies that the label of the nodes may change over time and the goal is to predict the correct label/category of the nodes at specific timestamps. It is important to note that in the dynamic node

Table 5.2: Full list of TGBase features.

	interaction-based		structure-based	
	intensity (w)	time (t)	neighborhood (N_u)	self (u)
target	$min(w_u^{in})$ $max(w_u^{in})$ $\overline{w_u^{in}}$ $\sum w_u^{in}$ $\sigma(w_u^{in})$ $H(w_u^{in})$	$min(t_u^{in})$ $max(t_u^{in})$ $\overline{t_u^{in}}$ $\sum t_u^{in}$ $\sigma(t_u^{in})$ $H(t_u^{in})$	$min(D_{in}(N_u))$ $max(D_{in}(N_u))$ $\overline{D_{in}(N_u)}$ $\sum D_{in}(N_u)$ $\sigma(D_{in}(N_u))$ $H(D_{in}(N_u))$	$D_{in}(u)$
source	$min(w_u^{out})$ $max(w_u^{out})$ $\overline{w_u^{out}}$ $\sum w_u^{out}$ $\sigma(w_u^{out})$ $H(w_u^{out})$	$min(t_u^{out})$ $max(t_u^{out})$ $\overline{t_u^{out}}$ $\sum t_u^{out}$ $\sigma(t_u^{out})$ $H(t_u^{out})$	$min(D_{out}(N_u))$ $max(D_{out}(N_u))$ $\overline{D_{out}(N_u)}$ $\sum D_{out}(N_u)$ $\sigma(D_{out}(N_u))$ $H(D_{out}(N_u))$	$D_{out}(u)$
either	$min(w_u^{tot})$ $max(w_u^{tot})$ $\overline{w_u^{tot}}$ $\sum w_u^{tot}$ $\sigma(w_u^{tot})$ $H(w_u^{tot})$	$min(t_u^{tot})$ $max(t_u^{tot})$ $\overline{t_u^{tot}}$ $\sum t_u^{tot}$ $\sigma(t_u^{tot})$ $H(t_u^{tot})$	$min(D_{tot}(N_u))$ $max(D_{tot}(N_u))$ $\overline{D_{tot}(N_u)}$ $\sum D_{tot}(N_u)$ $\sigma(D_{tot}(N_u))$ $H(D_{tot}(N_u))$	$D_{tot}(u)$

classification, we need to update node representations after observation of an event (such as evolving connectivity or features over time) to keep the most up to date representation. Moreover, in contrast to static node classification where node categories are predicted only once, in dynamic setting, classification can happen after each event to evaluate the impact of the network evolution on node categories. In dynamic node classification, after observation of each event, TGBase updates representation of the affected nodes to reflect the alteration. The update procedure for all the features is straightforward. For node classification, TGBase always utilizes the most up to date representation of the nodes based on the observed history so far.

This section presents the evaluation of TGBase for static and dynamic node classification task. We investigated eight real-world datasets presented in Table 5.3 in the context of two tasks, static (Section 5.2.3) and dynamic (Section 5.2.5) node classification. The first six rows of Table 5.3 provide information of the datasets for static node classification where the node classes are fixed, and the last two rows present the datasets utilized for dynamic node classification where node states may change over

Table 5.3: Statistics of the different benchmark datasets used for evaluation.

Dataset	Nodes	Edges	Benign nodes	Illicit nodes	Avg. degree	Node States	Network Category
Bitcoin	203,769	234,355	42,019	4,545	1.3002	static	cryptocurrency
Ethereum	2,973,489	13,551,303	2,972,324	1,165	9.1148	static	cryptocurrency
alpha	3,783	14,124	138	102	7.4671	static	cryptocurrency rating
OTC	5,881	21,492	136	180	7.3090	static	cryptocurrency rating
Amazon	330,317	560,804	2,358	241	3.3956	static	general rating
Epinions	195,805	4,835,208	9,291	1,013	42.7914	static	general rating
Reddit	10,000	672,447	10,634	366	15.6986	dynamic	social network
Wikipedia	8,227	157,474	8,010	217	4.4327	dynamic	social network

time. We explain the two settings and their corresponding datasets and baselines in the following sections.

5.2.3 TGBase for Static Node Classification.

Here, we evaluate TGBase for static node classification in temporal weighted networks.

Datasets. We consider the following available benchmark datasets:

• *Bitcoin.* We adopted Bitcoin transactions dataset presented by Weber et al. [45] that consists of more than $200K$ transactions of which 2% are labeled as *illicit* (corresponding to different categories of malicious activities including scams, ransomware, etc.), 21% are *licit* (corresponding to transactions of genuine categories including miners, exchanges, etc.), and there are no ground truth labels available for the rest of the transactions. Weber et al. [45] have also shared their defined set of features. The dataset is fully anonymized and the intensity and timestamp of the edges are not available. Therefore, we are only able to extract *structural* features (namely *self* and *neighborhood*). To preserve the semantic attributes of the nodes, we combined features provided by Weber et al. [45] with TGBase features.

• *Ethereum.* Transaction records of Ethereum network are presented by Wu et al. [85]. This dataset includes the transaction histories of different Ethereum accounts where 1165 accounts were involved in phishing activities. No ground truth labels are available for other nodes which are therefore assumed to be genuine.

• *OTC* and *Alpha.* These datasets consist of the user-to-user trust network of Bitcoin clients using OTC and Alpha platform for trading cryptocurrencies [120]. In these platforms, users who are anonymously trading Bitcoin can rate other members of the network based on their trustworthiness. Thus, fraudulent users have high monetary incentive for giving fake ratings. The ground truth in these networks are set based on the platform's founder rating. The founder and all other users he/she has rated highly

positive are considered as benign, and the users who have been negatively rated by the benign users are marked fraudulent.

• *Amazon.* This is a user-to-product rating network [120]. The helpfulness of a rating can be a good indicative of a fraudulent behavior, thus users with equal to or more than 50 votes, where the fraction of helpful-to-total votes is ≥ 0.75, are considered as genuine, while if the same fraction is ≤ 0.25 the user is considered as fraudulent.

• *Epinions.* This dataset consists of a user-to-post network in which rating values varies in range $[1, 6]$. A user-to-user trust network is used for defining the ground truth, where a user is labeled as fraudulent if its total trust rate is ≤ -10, and genuine if its trust rate is $\geq +10$ [120, 251].

Baselines. To make a comprehensive performance evaluation, we considered following baselines:

• *Network Embedding Baselines: node2vec* [148] *and RiWalk* [149]. We set the parameters of the node2vec in line with [148]: context size $k = 10$, embedding size $d = 64$, walk length $l = 5$, and number of walk per node $r = 20$. In addition, for better exploitation of the structural equivalency of the nodes, we set $p = 0.25$ and $q = 4$. We also used *RiWalk-WL* which captures the fine-grained similarities by imitating the neighborhood aggregation of the Weisfeiler-Lehman graph kernels [149].

• *GNN Baselines:* We compared the performance of TGBase against three GNN baselines which offer end-to-end node classification: *GCN* [154], *GraphSAGE* [155], and *GAT* [156]. The GNN baselines contain multiple layers where at each layer, the input is the node representations at that layer and the output is the transformed representations. At the final layer, the node representations are utilized for the classification task. We implemented all the GNN baselines in PyTorch Geometric with three layers and weighted loss function to considered the class imbalance of the datasets. We considered two different cases for initializing the node features for GNN baselines. In the first scenario, GNN models exploited one-hot encoding of the nodes as the initial features, which is a common practice. In the second scenario, TGBase is exploited for generating the initial features (indicated as *"TGBase → GNN"* in Table 5.4).

• *Platform-dependent Baselines.* We also compared the performance of TGBase with state-of-the-art (SOTA) baselines that are specifically designed for a particular type of network. These methods rely on particular characteristics of the network for which they are developed.

◦ *Cryptocurrency networks.* We considered the method proposed by Weber et al. [45] and trans2vec [85] as baselines for Bitcoin and Ethereum network, respectively. We set

Table 5.4: Comparing the performance of TGBase in terms of AUC with SOTA baselines in *static* node classification task. The **first**, **second**, and **third** best performing method are colored correspondingly.

Method	Bitcoin	Ethereum	Alpha	OTC	Amazon	Epinions
Platform-dependent	**0.955** ±0.006	0.827 ±0.010	0.833 ±0.112	0.873 ±0.104	0.847 ±0.155	0.856 ±0.112
GCN	0.545 ±0.056	0.556 ±0.069	0.729 ±0.042	0.697 ±0.036	0.610 ±0.011	0.534 ±0.071
GraphSAGE	0.553 ±0.054	0.647 ±0.145	0.624 ±0.101	0.681 ±0.114	0.571±0.051	0.560 ±0.069
GAT	0.582 ±0.005	0.576 ±0.103	0.732 ±0.081	0.836 ±0.033	0.513 ±0.019	0.521 ±0.031
TGBase → GCN	0.901 ±0.007	0.792 ±0.020	0.912 ±0.023	0.942 ±0.019	0.934 ±0.020	0.859±0.011
TGBase → GraphSAGE	**0.970** ±0.003	**0.958** ±0.010	0.984 ±0.003	0.983 ±0.004	0.930±0.012	**0.950** ±0.008
TGBase → GAT	0.928 ±0.010	0.709 ±0.041	0.926 ±0.024	0.948 ±0.012	0.926 ±0.015	0.555 ±0.156
node2vec	0.669 ±0.012	0.817 ±0.026	**0.990** ±0.003	**0.996** ±0.001	**1.000** ±0.000	*nc*
RiWalk + RF	0.636 ±0.050	0.805 ±0.022	**0.991** ±0.003	**0.995** ±0.002	**1.000** ±0.000	*nc*
TGBase	**0.953** ±0.006	**0.906** ±0.010	**0.999** ±0.001	**1.000** ±0.000	**1.000** ±0.000	**0.999** ±0.000
node2vec	0.636 ±0.020	0.793 ±0.026	0.716 ±0.071	0.784 ±0.046	**0.999** ±0.001	*nc*
RiWalk + MLP	0.626 ±0.055	0.722 ±0.036	0.681 ±0.068	0.770 ±0.042	**0.999** ±0.001	*nc*
TGBase	0.940 ±0.008	**0.892** ±0.014	0.764 ±0.102	0.892 ±0.0.052	**0.936** ±0.017	**0.932** ±0.010

the parameters of the trans2vec similar to those of node2vec for a fair comparison. Please note that these methods cannot be applied to a different domain or even cryptocurrency platform. For example, trans2vec is not applicable to Bitcoin and is defined only for Ethereum.

- *Cryptocurrency rating and general rating networks.* We considered REV2 [120], which is the SOTA method for detecting fraudsters in rating benchmark datasets.

Performance Evaluation. In case of Bitcoin and Ethereum networks, we considered the malicious nodes and an equal number of genuine nodes as the set of anchor nodes and then, extracted the first-order neighbors of the anchor nodes and all the edges amongst them to extract subgraphs from the original networks. We repeated the random genuine anchor nodes selection procedure 10 times, and reported the average performance. For the rating networks, namely Alpha, OTC, Amazon, and Epinions we considered the classification task to be applied on all labeled nodes, while the available graphs are utilized in generating node representations. We reported the average performance over 10 different iterations of random train-test splits. The performance is measured using *area under the ROC curve (AUC)*.

After generating node representations with TGBase, we need a classifier for predicting the category of the nodes. We considered two different classifiers. Namely, we implemented a three-layered *Multi-Layer Perceptron (MLP)* classifier with a ReLU activation function in PyTorch. We also exploited *Random Forest (RF)* due to its high performance [45, 120]. RF (with the following setup: number of estimator = 50,

maximum number of features = 10, and maximum depth = 5) was implemented using Scikit-learn Python package. The evaluation results are illustrated in Table 5.4.

Results. We discuss the results of each dataset in the following.

• *Bitcoin.* The results in the second column of Table 5.4 show that both generic node embedding methods and GNNs (with one-hot encodings) achieve a relatively low performance on this benchmark. This is expected since these models do not deal with the intrinsic characteristics of financial networks, and mainly emphasize on the structural similarities to gain the node embeddings. However, TGBase which is also a generic method performs competitively to the platform dependent contender on this dataset. This can be attributed to the fact that TGBase takes into account important characteristics of the network mainly the interaction intensity and timestamp, which are overlooked by the baselines.

• *Ethereum.* The results in the third column of Table 5.4 suggest that TGBase significantly outperforms the baselines in classification of the malicious nodes on the Ethereum network. Notably, TGBase shows higher performance compared to trans2vec that is specifically designed for the Ethereum network. The generic node embedding methods also give a relatively good performance on this dataset which suggests the structural information are important. The GNNs, when do not exploit TGBase features, are however still underperforming on this dataset.

• *Alpha* and *OTC.* The results in the fourth and fifth column of Table 5.4 represent the higher performance of TGBase compared to the baselines for OTC and Alpha datasets, respectively. Again, TGBase achieves better performance compared to REV2, the domain specific SOTA, demonstrating that TGBase's representations are more suitable for distinguishing malicious nodes. It is important to note that the size of the Alpha and OTC are much smaller than the size of the Bitcoin and Ethereum networks. This might explain why node2vec and RiWalk show better performacne on Alpha and OTC networks, compared to bigger networks. TGBase can achieve good performance regardless of the network size.

• *Amazon.* The evaluation results are presented in the sixth column of Table 5.4. We observe a similar pattern where more powerful or complex models have lower performance, while TGBase achieves a perfect results. It should be noted that labeled nodes in the Amazon dataset are extremely imbalanced which can explain the higher AUC values.

• *Epinions.* The results on Epinions dataset is presented in the seventh column of Table 5.4. Due to the huge size of Epinion network, node2vec and RiWalk did not converge in reasonable time. However, we can observe that TGBase reaches significantly better

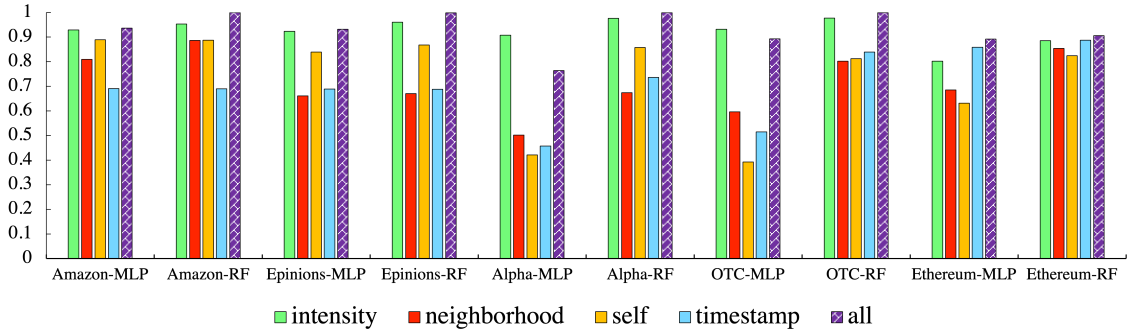


Figure 5.2: Results of ablation study on the performance of different sets of TGBase features in the static node classification task, where the reported metric is *AUC*.

performance compared to REV2 [120], which implies the effectiveness of the proposed feature set on this dataset.

Overall, TGBase achieves the best perfomance across different networks, and GNN methods (in case they did not employ TGBase for generating node initial features) mostly achieve lower performance on all datasets compared to other methods. There are two reasons that can explain the lower performance of GNN methods when using one-hot encoding as initial features. First, GNN methods considerably depend on the initial features of the nodes, however, the considered datasets do not provide any initial node features. Second, GNN methods performs end-to-end node classification where they optimize the loss function to achieve high accuracy. However, these datasets are highly unbalanced and high accuracy can be achieved even by incorrectly predicting all instances as negative. Thus, although the methods can achieve high accuracy, we can observe that other performance measures, including AUC, which are more proper for unbalanced datasets, indicate a low performance when initial features are not expressive enough.

5.2.4 Impact of Different Groups of TGBase features.

To better evaluate the impact of different TGBase's sets of features, we conducted node classification when only one set of TGBase features are used. Particularly, we evaluate the node classification performance, when only intensity, neighborhood, self, or timestamp features are employed as the node representations. As shown in Fig. 5.2, the aggregation of all features results in the best overall performance. Moreover, comparing the classification performance of individual feature sets, we can observe that intensity features can mostly achieve the best results, which implies the importance of

incorporating the edge-weight based attributes in the node representations. Note that the Bitcoin dataset is anonymized and we could not define all sets of features for nodes of Bitcoin network, therefore this dataset is dropped here.

5.2.5 TGBase for Dynamic Node Classification.

This section elaborates on evaluating the performance of TGBase for dynamic node classification task against strong baselines that are specifically designed for representation learning on temporal graphs.

Datasets. For the experiments, we consider *Reddit* and *Wikipedia* datasets consisting of timestamped interactions of users on these social networks with evolving node labels corresponding to their reputation in the network. The goal of the classification task is to predict whether the state of a user will change due to an interaction. Specifically, for normal users that are not banned from posting sub-Reddits or editing Wikipedia pages, their label is always '0', while the label of a banned user changes to '1' after its final interaction.

• *Reddit.* This dataset consists of a bipartite graph including $1,000$ most active posts made by $10,000$ most active users on sub-Reddits during a period of one month [218]. The interactions among users and sub-Reddits are associated with text attributes representing the text information of the posts.

• *Wikipedia.* The dataset consists of a bipartite graph representing the edits made by users on Wikipedia pages. The $1,000$ most edited pages together with the users who made a minimum of 5 edits are considered as the nodes, and an edge demonstrates a user editing a page [218]. The interactions are associated with text attributes related to the page edits. Similar to the Reddit dataset, the edge weights of the Wikipedia network also consist of vectors of attributes; thus, the intensity features of TGBase are defined for each attributes of the edge weight vectors. Node labels and interactions are timestamped, and nodes are labeled based on their states which represent whether a user is banned from editing a page.

Baselines. For dynamic node classification task, we compared TGBase with four SOTAs on representation learning on temporal networks including JODIE [218], DyRep [219], TGAT [220], and TGN [221]. According to node interaction timestamps, we did a chronological train-validation-test split with 70%-15%-15% in line with the baseline methods [218, 219, 220, 221].

• *JODIE* [218] models the interactions of the users and items in domains such as

Table 5.5: Comparing the performance of TGBase in terms of AUC with SOTA baselines in *dynamic* node classification task. The **first**, **second**, and **third** best performing method are colored correspondingly.

Method	Wikipedia	Reddit
JODIE [2019]	0.862 (±0.004)	0.675 (±0.006)
DyRep [2019]	0.837 (±0.007)	**0.691** (±0.005)
TGAT [2020]	0.501 (±0.001)	0.502 (±0.001)
TGN [2020]	**0.881** (±0.001)	0.668 (±0.008)
TGBase + RF	**0.874** (±0.010)	**0.713** (±0.007)
TGBase + MLP	**0.882** (±0.004)	**0.730** (±0.001)

social networks. JODIE utilizes RNNs to update the representations of the source and target nodes every time an interaction takes place.

- *DyRep* [219] intends to consider the topological evolution, as well as activities between the nodes to capture the dynamics of the interactions in the networks.
- *TGAT* [220] aims to aggregate temporal and topological features to recognize the node representations as function of time. It leverages GAT as its building blocks and develops functional time encoding
- *TGN* [219] is an efficient and generic framework that combines graph-based operators and memory modules for representation learning on temporal graphs.

Performance Evaluation. For Reddit and Wikipedia datasets, we predict the dynamic node labels, as related to the reputation ranking. Particularly, the downstream dynamic node classification task is used to predict whether the user is banned. The results are illustrated in Table 5.5.

Results. In Table 5.5, we can see that TGBase obtains state-of-the-art results on both benchmark datasets. On Wikipedia, results are on par with TGN, whereas TGBase significantly outperforms the baselines on Reddit dataset. We want to reemphasize that the features we extract require zero-learning or parameter adjustment. Yet, coupled with a shallow classifier, they are outperforming SOTA methods across different domains and settings.

5.3 On Detection of Anomalies in Graphs

While investigating different state-of-the-art baseline methods, we observe that there is increasing usage of anomaly detection datasets in the node classification task with a balanced setting. Specifically, many of the datasets that are used for the evaluation of the node classification methods contain anomalous instances. Since anomalies are essentially exceptional instances, there are very rare, which results in a high imbalance of these datasets. Therefore in this section, we are motivated to investigate the node classification settings for the anomaly detection task.

In the present epoch of big data, many real-world phenomena can be explored and represented through the unifying abstractions offered by graphs. In many diverse and complex data exploration and management ecosystems, big graphs processing has emerged as a principal computing framework with applications in many domains including security, social networks, finance, and many more [252]. Considering that in many usecases the big data consists of relations, as well as vectors of features, a vital challenge is to leverage information embedded in interconnected data that is modelled by graphs.

The most universal data structures that can be leveraged for extracting information from complex relational structures are graphs that are deployed in many different applications, such as financial networks (e.g., graph of cryptocurrency transactions, supply chain graph), rating networks (e.g., user-to-user rating platforms, user-to-product rating graph), social networks (e.g., relations represented as friendship between users, message, or email), etc [147]. Due to the importance of the additional valuable information provided by the relations of the entities in networks, graph-based methods are emerging as the mainstream approaches in industrial applications involving relational information [142]. Recently, investigation of graphs via machine learning approaches has witnessed a great surge of interest [147] in myriad of domains including social science [253], knowledge graphs [254], and finance [20]. The impact of employing efficient machine learning algorithms for big graph analysis is observable in mitigating important and complex problems such as alleviating the current pandemic through analytic offered by *Graph 4 COVID-19* initiative [252, 255].

One main category of tasks in network analysis is node classification [148, 155]. The task of node classification involves classifying each node of a network into one of predefined sets of classes [142]. When modeling node classification as a supervised machine learning task, the node representations can be employed by any off-the-shelf machine learning

classifier to predict the classes of the nodes [148]. Therefore, the representations of the nodes can be considered as a vector that efficiently encodes information about each node's neighborhood into a feature vector which can be exploited in different downstream tasks [155].

Node classification can be also employed for detecting anomalous entities on networks. For instance, detecting malicious users in financial networks [20], detecting fraudulent users in rating platforms that give fake rating for monetary outcomes [120], or spammer in social or financial networks [256] have all been modeled as machine learning tasks where the ultimate objective is to efficiently predict the node classes using node representations as feature vectors. In fact, it can be observed that many of the datasets that are extensively being used in node classification tasks contains one class of nodes that is associated with anomalous activities [45, 85, 120, 218]. Hence, the node classification task on these networks are indeed a supervised anomaly detection task.

Essentially, anomalous instances refer to those that considerably deviate from seemingly normal observations [16]. In order to prevent various detrimental events such as scams and frauds in financial transaction networks or social spams, a vital task is the detection of anomalous instances. One important challenge in anomaly detection is the insufficiency of labelled data, which leads the main approaches for anomaly detection towards unsupervised methods or rule-based heuristic approaches [10]. Although most of classification methods address the problem in a relatively balanced setting, real-world scenarios often present datasets where some classes have considerably fewer number of instances. Training the classifiers unaware of the of the intrinsic imbalance of the datasets may results in under-representation of instances from the minority class and consequently, sub-optimal performance of the classification task [257]. In this work, we explore the application of supervised methods for classification of unbalanced datasets and demonstrate the importance of considering the intrinsic imbalance of the instances. Specifically, we focus on datasets consisting of anomalous and normal instances and investigate the performance of various node embedding and classification approaches for supervised classification task.

Imbalance problem is one of the greatest issues in data mining which relates to the case that one of the classes have considerably less number of the instances compared to others [136]. The classification methods, if overlooking the imbalance issue, mostly focus on the samples from the majority class and aim to optimize classification accuracy, while ignoring or misclassifying minority samples [136]. This becomes a vital drawback

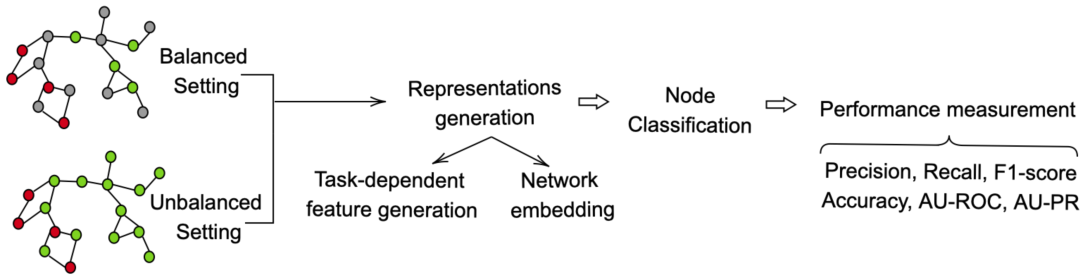


Figure 5.3: Overview of the experimental pipeline. Red and green nodes respectively denote anomalous and normal nodes, while the nodes that are not included in the balanced setting classification are colored in gray.

when applying classification for anomaly detection. For one thing, the datasets including anomalies are extremely unbalanced which results in poor performance of the methods. For another thing, although the minority samples are very rare, there are extremely important to be detected and predicting false negatives could be very costly. Examples of which include credit card fraud detection or detecting faults in safety critical systems [18, 136].

In this study, we demonstrate the importance of considering the class imbalance in supervised anomaly detection when using node classification techniques. This study is motivated by the increased usage of anomaly detection datasets in node classification works [45, 85, 120, 249], as well as many recent works ignoring the class imbalance issue. We employ various node embedding methods including task-dependent and structural network embedding for generating node representations, while employing several classifiers for the downstream node classification task. We evaluate the performance of various approaches in two different settings namely *balanced* and *unbalanced* which are defined based on the distribution of the minority anomalous class. We investigate various evaluation metrics in either setting to thoroughly contrast the characteristics of different settings and provide recommendations for choosing practical strategies when dealing with unbalanced datasets.

5.3.1 Anomaly Detection in Graphs as Node Classification

In this work, we focus on assessment of leveraging node classification approaches for anomaly detection in real-world networks. Given a large network of entities, where only a small portion of them are labelled as being associated with anomalous activities, the goal is to evaluate the performance of different node representations working jointly with

classification methods for detection of the anomalous instances. In addition, we are interested to investigate the effectiveness of different evaluation metrics in demonstrating the performance of different approaches. We basically consider two different settings, namely *balanced* and *unbalanced*, for applying node classification. In the balanced setting, we focus on the classification of a balanced set of instances including all anomalous nodes and an equal number of normal nodes. In contrast, in the unbalanced setting, the classification is applied to all the available nodes of the networks. The performance of the node classification task in each setting is inspected based on different evaluation metrics and several recommendations from a practitioner's perspective are provided. An overview of the experimental assessment pipeline is demonstrated in Fig. 5.3. As shown, for both of the the setting, different sets of experimental assessments are performed. Mainly, we consider generating node representations by a task-dependent or a general network embedding method. The generated node representations are then employed by several classifiers for detection of anomalous instances. At the end, the performance of the downstream task are evaluated with different performance metrics. Our main objective is the assessment of different settings, node representations, and classification methods with a comprehensive sets of performance metrics for the anomaly detection problem. The outcomes of the assessment are represented as several recommendations that can be helpful to the practitioners when applying node classification methods for detecting anomalies in large graphs.

Datasets

We assess the performance of different node representations and classification methods on six real-world datasets (with static node labels) whose statistics are presented in Table 5.3. These datasets (specifically, *Bitcoin*, *Ethereum*, *OTC*, *Alpha*, *Amazon*, *Epinions*) are basically examples of real-world networks in which a small portion of entities are labelled as being associated with different malicious activities, while the majority of the nodes are assumed to be normal. These datasets are explained in more detail in Section 5.2.3.

5.3.2 Experimental Analysis

One of our main goal is to provide a comparative perspective of the two different settings (namely *balanced* and *unbalanced*) that are currently used by the practitioners for the anomaly detection problem. Considering that anomaly detection has several important usecases, it is important to have an efficient combination of an evaluation setting and

Table 5.6: Performance evaluation on cryptocurrency networks.

			Balanced *Setting*						***Unbalanced*** *Setting*					
		Algorithm	Precision	Recall	F_1	Accuracy	AU-ROC	AU-PR	Precision	Recall	F_1	Accuracy	AU-ROC	AU-PR
Bitcoin	*RF*	Weber et al. [45]	0.987	0.845	0.911	0.911	0.982	0.986	0.923	0.687	0.788	0.921	0.955	0.888
		node2vec	0.804	0.378	0.513	0.617	0.684	0.745	0.710	0.119	0.203	0.806	0.602	0.379
		RiWalk	0.632	0.364	0.453	0.540	0.582	0.647	0.728	0.108	0.189	0.806	0.635	0.397
	LR	Weber et al. [45]	0.918	0.906	0.912	0.906	0.972	0.974	0.824	0.700	0.757	0.905	0.940	0.777
		node2vec	0.694	0.285	0.399	0.548	0.579	0.662	0.813	0.150	0.254	0.816	0.652	0.472
		RiWalk	0.666	0.261	0.367	0.533	0.574	0.643	0.735	0.082	0.146	0.801	0.635	0.393
	MLP	Weber et al. [45]	0.934	0.861	0.896	0.892	0.961	0.966	0.730	0.855	0.787	0.902	0.949	0.873
		node2vec	0.732	0.435	0.480	0.573	0.596	0.665	0.315	0.540	0.393	0.646	0.624	0.387
		RiWalk	0.745	0.245	0.358	0.540	0.571	0.641	0.360	0.472	0.408	0.706	0.634	0.391
Ethereum	*RF*	trans2vec	0.888	0.973	0.928	0.920	0.961	0.958	0.110	0.002	0.004	0.956	0.829	0.198
		node2vec	0.891	0.972	0.930	0.922	0.967	0.964	0.488	0.039	0.072	0.956	0.815	0.212
		RiWalk	0.918	0.932	0.925	0.920	0.934	0.939	0.100	0.000	0.001	0.956	0.805	0.178
	LR	trans2vec	0.911	0.910	0.911	0.905	0.958	0.958	0.276	0.073	0.115	0.951	0.783	0.178
		node2vec	0.915	0.932	0.923	0.918	0.964	0.959	0.286	0.146	0.192	0.946	0.799	0.173
		RiWalk	0.921	0.852	0.885	0.882	0.932	0.932	0.120	0.009	0.017	0.951	0.726	0.095
	MLP	trans2vec	0.890	0.970	0.928	0.921	0.955	0.952	0.097	0.730	0.170	0.676	0.761	0.164
		node2vec	0.887	0.973	0.928	0.920	0.955	0.953	0.103	0.836	0.183	0.665	0.797	0.160
		RiWalk	0.912	0.899	0.905	0.900	0.913	0.911	0.093	0.761	0.165	0.655	0.723	0.100

performance metrics in order to better inspect different methods. To this end, we have employed node representation learning methods including task-dependent and network embedding for generating node representations for binary classification of the nodes to anomalous versus normal ones.

5.3.3 Evaluation Settings

We focused on assessing the performance of node classification approaches for anomaly detection in two distinct settings. Having in mind the high imbalance nature of datasets containing anomalies, we considered the following settings.

Balanced Setting. Since imbalance issue of the datasets makes several challenges for the classification task which may result in its poor performance, in this setting, we intended to eliminate the dataset imbalance through under-sampling of the normal nodes with the goal of attaining a roughly balanced dataset for the classification task. Particularly, we considered all the available nodes of the network in the node representation learning procedure. Therefore, when generating node representations through task-dependent or network embedding methods, no information is lost, and these methods can exploit all the available structural and content-related information. However, in the classification phase, we under-sampled the normal nodes in such a way to have a balanced dataset. Thus, the under-sampling happened in the feature space. For balancing the dataset, we preserved all the anomalous nodes, and randomly selected similar number of normal nodes to produce the set of node features that is used by the

classifier. We repeated the random selection procedure of the normal nodes in 10 different runs and reported the average performance.

Unbalanced Setting. This is the conventional setting used in majority of the node classification approaches where all nodes are considered in representation learning as well as in the classification procedure. The advantage of this setting is that all available information is exploited for the classification. However, since the distribution of the classes in datasets containing anomalous instances is highly skewed, the performance of the classifier can be severely damaged.

In each of the two different settings, first the node representations were generated by either a task-dependent method (such as Weber et. al. [45] for Bitcoin, trans2vec [85] for Ethereum, or REV2 [120] for cryptocurrency rating and general rating networks) or a general network embedding technique (such as node2vec [148] or RiWalk [149]). Then, the node representations were leveraged by a binary classifier (such as Random Forest (RF), Logistic Regression (LR), or Multi-Layer Perceptron (MLP)) for the detection of the anomalous nodes. Different methods and their implementation details are elaborated as follows.

• ***Task-dependent node representation approaches***. These approaches mainly provide meaningful feature vectors for each node of the network considering the characteristics of the nodes on a specific network. Particularly, the feature sets are specifically designed for each task based on the attributes and content of the application as well as the underlying network.

- **Weber et. al.** [45]. For the Bitcoin network dataset, we considered the feature set provided by Weber et. al. [45] as the node representations for the downstream classification task. The proposed feature set consisted of 94 features which expressed the local information about each transaction node (e.g., the timestamp and transaction fee) and 72 aggregated features gained by aggregating information from direct neighbors of each node.

- **Trans2vec** [85]. In case of the Ethereum transaction network, we employed trans2vec [85] for generating the node representations for Ethereum accounts. Trans2vec is a random walk-based node embedding method that exploited the timestamp and amount of transactions in edge weight generation which were then used to direct the selection of nodes in random walks. The parameters of the trans2vec were set inline with those of node2vec for a fair comparison.

- **REV2** [120]. Regarding the rating platforms, namely cryptocurrency rating and

general rating, we adopted the node representation learning approach proposed by Kumar et. al. [120] which is the state-of-the-art approach for the task of detecting fraudulent users in rating platforms. REV2 leveraged an iterative process where the network information as well as the behavioral properties were used for generating the node representations.

• ***Network embedding approaches***. Another group of node representation methods that we have exploited for gaining node features were two state-of-the-art shallow network embedding methods namely node2vec and RiWalk that are random walk-based methods generating node representations in an unsupervised manner. In line with the initial paper proposing node2vec [148, 149], we set the parameters of node2vec and RiWalk as follows: walk length $l = 5$, number of walks per node $r = 20$, embedding size $d = 64$, context size $k = 10$, and $p = 0.25$ and $q = 4$ for better exploitation of the structural equivalency of the nodes.

• ***Binary classifiers***. Gaining the node representations from the aforementioned approaches, we utilized three different classifiers for the downstream node classification task. We tested Random Forest (RF), Logistic Regression (LR), and Multi-Layer Perceptron (MLP) as our supervised classifiers. The implementation details of the classifiers are as follows: RF with number of estimator = 50, maximum number of features = 10, and maximum depth = 5, and LR with *L1* regularization were implemented using Scikit-learn Python package. The MLP was implemented in PyTorch with three layers and ReLU activation function.

The results of node classification in balanced and unbalanced setting are illustrated in Table 5.6, Table 5.7, and Table 5.8 for cryptocurrency, cryptocurrency rating, and rating network, respectively. These tables represent the average performance of different settings among 10 different runs. The standard deviations corresponding to the results reported in Table 5.6, Table 5.7, and Table 5.8 are illustrated in Table 5.10, Table 5.11, and Table 5.12 in Appendix 5.4, respectively.

5.3.4 Results Analysis and Recommendations

Considering the experimental results of node classification in two different setting, namely balanced and unbalanced, which are represented in Table 5.6, Table 5.7, and Table 5.8, we can make several observations. First, it can be observed that task-dependent node representations approaches mostly generate more efficient representations that help in achieving higher performance of node classification in both balanced and unbalanced

Table 5.7: Performance evaluation on cryptocurrency rating networks.

			***Balanced** Setting*						***Unbalanced** Setting*					
	Algorithm		Precision	Recall	F_1	Accuracy	AU-ROC	AU-PR	Precision	Recall	F_1	Accuracy	AU-ROC	AU-PR
Alpha	*RF*	REV2	0.827	0.645	0.702	0.771	0.818	0.723	0.802	0.659	0.698	0.746	0.815	0.701
		node2vec	0.672	0.629	0.644	0.698	0.761	0.710	1.000	0.000	0.000	0.973	0.571	0.031
		RiWalk	0.681	0.650	0.660	0.708	0.782	0.718	1.000	0.000	0.000	0.973	0.602	0.042
	LR	REV2	0.785	0.671	0.708	0.758	0.833	0.762	0.611	0.532	0.569	0.879	0.788	0.711
		node2vec	0.714	0.622	0.658	0.715	0.804	0.745	1.000	0.000	0.000	0.973	0.611	0.051
		RiWalk	0.698	0.590	0.630	0.701	0.798	0.759	1.000	0.000	0.000	0.973	0.596	0.047
	MLP	REV2	0.883	0.594	0.708	0.756	0.820	0.646	0.689	0.469	0.558	0.847	0.887	0.464
		node2vec	0.690	0.743	0.698	0.724	0.736	0.657	0.039	0.641	0.073	0.552	0.600	0.036
		RiWalk	0.706	0.740	0.701	0.724	0.747	0.671	0.054	0.519	0.085	0.630	0.572	0.039
OTC	*RF*	REV2	0.818	0.858	0.838	0.823	0.897	0.835	0.631	0.672	0.651	0.879	0.922	0.598
		node2vec	0.779	0.853	0.823	0.789	0.829	0.828	0.600	0.021	0.041	0.969	0.740	0.109
		RiWalk	0.776	0.816	0.794	0.758	0.801	0.802	0.800	0.027	0.052	0.970	0.737	0.110
	LR	REV2	0.802	0.787	0.794	0.801	0.873	0.842	0.596	0.643	0.619	0.833	0.855	0.687
		node2vec	0.790	0.760	0.772	0.742	0.819	0.838	0.100	0.003	0.006	0.969	0.740	0.111
		RiWalk	0.767	0.744	0.753	0.721	0.788	0.805	1.000	0.000	0.000	0.969	0.728	0.106
	MLP	REV2	0.835	0.722	0.774	0.788	0.766	0.718	0.704	0.568	0.635	0.877	0.852	0.547
		node2vec	0.778	0.795	0.774	0.736	0.715	0.728	0.067	0.660	0.122	0.703	0.675	0.056
		RiWalk	0.749	0.767	0.724	0.692	0.642	0.689	0.062	0.619	0.112	0.694	0.648	0.056

setting. This is mainly because of the fact that these methods incorporate the extra information available in the dataset as edge weights or edge timestamps. While the general network embedding methods (such as node2vec and RiWalk) do not generally leverage edge features, task-dependent methods define their sets of features according to the intrinsic characteristics of the networks and their contents. Therefore, although their application is tailored to a specific task and they cannot be directly extended to other applications, they mostly show higher performance on their specific platform compared to more general approaches.

Moreover, we can observe that different performance metrics demonstrate different characteristics in balanced and unbalanced setting. For example, while high value of accuracy is observed in both settings, other performance measures, like the precision and recall, that are more commonly adopted in datasets with imbalance issue tell a different story about the performance of the approaches in these two settings. Specially in unbalanced setting, methods can predict all instances as negative and still achieve a high accuracy value. However, as detection of the positive instances in the anomaly detection task is of great importance, this scenario is not appealing and infers the necessity of better performance metrics.

For better comparison of the balanced and unbalanced setting, we have presented the correlation and average difference (i.e., *balanced. perf.* − *unbalanced. perf.*) of each performance metric in the two setting in Table 5.9. We have also demonstrated the correlation of various performance metrics in either balanced or unbalanced setting in

Table 5.8: Performance evaluation on rating networks. **NC* denotes that the node representation approach did not converged in reasonable time (we set the time limit as two days). Thus, we were not able to gain the node embeddings for the classification task.

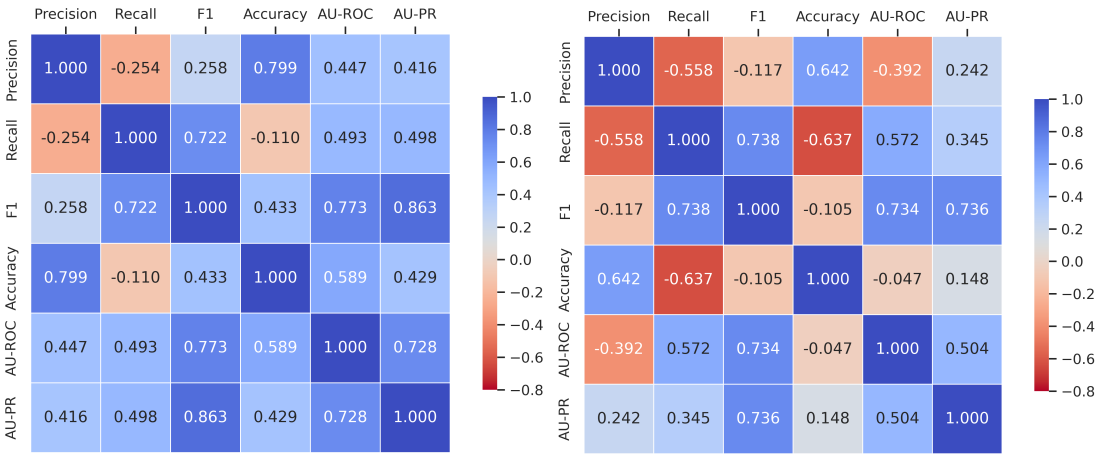
			***Balanced** Setting*						***Unbalanced** Setting*					
	Algorithm		Precision	Recall	F_1	Accuracy	AU-ROC	AU-PR	Precision	Recall	F_1	Accuracy	AU-ROC	AU-PR
Amazon	*RF*	REV2	0.801	0.811	0.806	0.804	0.854	0.678	0.587	0.623	0.604	0.919	0.921	0.536
		node2vec	0.501	0.509	0.502	0.484	0.470	0.491	1.000	0.000	0.000	0.999	0.480	0.001
		RiWalk	0.551	0.606	0.576	0.538	0.545	0.562	1.000	0.000	0.000	0.999	0.519	0.001
	LR	REV2	0.802	0.799	0.800	0.807	0.847	0.841	0.647	0.591	0.618	0.927	0.914	0.657
		node2vec	0.519	1.000	0.683	0.519	0.500	0.759	1.000	0.000	0.000	0.999	0.500	0.500
		RiWalk	0.519	1.000	0.683	0.519	0.500	0.759	1.000	0.000	0.000	0.999	0.500	0.500
	MLP	REV2	0.816	0.678	0.739	0.769	0.811	0.789	0.624	0.657	0.633	0.837	0.807	0.769
		node2vec	0.558	0.462	0.440	0.534	0.464	0.505	0.001	0.711	0.002	0.345	0.473	0.001
		RiWalk	0.589	0.606	0.543	0.549	0.487	0.515	0.001	0.687	0.002	0.367	0.485	0.001
Epinions	*RF*	REV2	0.821	0.768	0.794	0.863	0.877	0.873	0.576	0.498	0.534	0.944	0.896	0.655
		node2vec	NC*	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC
		RiWalk	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC
	LR	REV2	0.769	0.758	0.764	0.841	0.856	0.857	0.503	0.434	0.466	0.895	0.887	0.699
		node2vec	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC
		RiWalk	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC
	MLP	REV2	0.834	0.713	0.771	0.827	0.892	0.773	0.648	0.597	0.621	0.898	0.896	0.624
		node2vec	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC
		RiWalk	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC

Fig. 5.4. Considering the results, we can make the following recommendations.

- **Recommendation 1:** ***The evaluation setting should be consistent with the task's objective.***
  The evaluation setting when detecting anomalies by a node classification approach is important and the performance in the balanced setting does not correlate closely with the performance in unbalanced setting as shown in Table 5.9. Therefore, it is important to note that when evaluating different approaches for datasets containing anomalous samples, selecting the right setting is of paramount

Table 5.9: Comparing the correlation and average difference of various evaluation metrics in balanced and unbalanced setting. For computing performance differences, the value of a metric in the unbalanced setting is deducted from its counterpart in the balanced setting.

	Precision	Recall	F_1	Accuracy	AU-ROC	AU-PR
Correlation	-0.216	0.244	0.421	0.017	0.785	0.366
Average Difference	0.072	0.388	0.386	-0.165	0.061	0.355



(a) Balanced setting. **(b)** Unbalanced setting.

Figure 5.4: Correlation of different evaluation metrics in balanced or unbalanced setting.

importance. Particularly, different approaches should be evaluated in the setting that they are designed for or expected to perform in, otherwise the evaluation of their performance may be misleading. Indeed, we suggest to investigate the performance of node classification tasks in balanced setting, while evaluating the performance of anomaly detection tasks in unbalanced setting; or considering evaluation in both setting for a general purpose approach. This is inline with the fact that the real-world datasets for the anomaly detection tasks are extremely unbalanced, which is highly important to be considered when designing an anomaly detection method.

- **Recommendation 2:** ***The appropriate performance metric should be selected in line with the task under study.***
  In unbalanced setting, accuracy and AU-ROC are not appropriate performance metrics. As it is shown in Table 5.6, Table 5.7, and Table 5.8, classification tasks achieved high accuracy and AU-ROC in most of the cases. Additionally, we can see in some cases, the accuracy and AU-ROC of the unbalanced setting is even higher than the balanced setting, while obviously the classification task is more challenging in unbalanced setting due to high data imbalance. It is important to note that in the unbalanced setting, even if the classifier predicts every instances as negative, it can reach high accuracy, which again underlies the fact that accuracy is not an appropriate performance measure in unbalanced setting, and other performance measure should be considered for performance evaluation of

different approaches. Most notably, we observe a negative correlation between AU-ROC and precision in the unbalanced setting, as it is indicated in Fig. 5.4 and Table 5.9. It has been shown that compared to the accuracy, precision recall (and F_1-*score* that combines these latter two) are more informative performance metrics for the imbalance classification problem because they focus on the prediction of the positive instances [45, 248]. Hence, we are interested in achieving higher values for these two metrics. However, the negative correlation of AU-ROC and precision in the unbalanced setting infers that AU-ROC is not a reliable metric when the dataset is highly unbalanced.

- **Recommendation 3:** ***Evaluation of the node classification methods on anomaly detection datasets could cause results misinterpretation.***
  For the anomaly detection task, the balanced setting is overestimating the performance as shown in Table 5.6, Table 5.7, and Table 5.8. Essentially, most datasets presented for the anomaly detection task are extremely unbalanced and detecting anomalies in these datasets is very important. Evaluating an approach in balanced setting although may results in better performance of the classification task, is far from the actual setting in anomaly detection problems where normal samples outnumber anomalous ones. In addition, the difference of the performance between these two settings could be considerable as shown in Table 5.9. Hence, performance metrics in balanced and unbalanced setting do not always show high positive correlation (e.g., note that the correlation of precision in balanced and unbalanced setting is negative as shown in Table 5.9), which implies that reaching good performance in balanced setting does not necessarily results in good performance in the unbalanced setting as well.

5.3.5 Conclusion

In this chapter, we first proposed TGBase, a simple yet powerful method for node classification in weighted temporal graphs. TGBase encodes each node by extracting a small set of features based on the structural attributes of the node and its neighbors, as well as the intensity and timestamp attributes of the interactions among node pairs. Through extensive set of experiments we show that our simple shallow model outperforms more complex models which are currently the state-of-the-art in the static and dynamic node classification task. Moreover, our method is more general compared to these methods, as they are defined per specific datasets or a class of datasets.

TGBase is therefore *generic* and generates efficient node representations for all the available benchmark datasets that we know of. Given its low time and model complexity, our proposed TGBase is perfectly suited as a baseline when designing models for node classification in temporal graphs.

Furthermore, throughout Section 5.3, we explored the exploitation of node classification methods for anomaly detection in the context of real-world large graphs. Since an important challenge in the era of big data is to leverage the information as effectively as possible, graph-based techniques have emerged as leading approaches in different applications aiming to exploit the extra information available in relational data. One important challenge in big data and graph analysis is the existence of anomalous patterns in wide range of disciplines. Hence, anomaly detection is amongst the vital tasks in network analysis whose performance is principal in preventing adverse situations like financial frauds and social spams. In this thesis, we assessed the performance of node classification for an anomaly detection task in balanced and unbalanced setting. We investigated different performance metrics in our evaluation and showed that the tasks, settings, and performance metrics should be selected in accordance with the intrinsic characteristics of the datasets and usecases. Based on our extensive assessments, we made several recommendations that could help practitioners to better decided about the experimental settings when resolving a node classification or anomaly detection problem on large networks. As a future direction, it is interesting to investigate the performance of other representation learning methods. Important examples of these methods include graph neural networks that jointly generate node representations and classify the instances, and are trained end-to-end. Considering the training procedure of these methods being aware of the dataset imbalance is another challenging future work.

5.4 Appendix: Additional Results

The additional results related to the ones reported throughout Section 5.3.3 are presented.

Table 5.10: The standard deviation of 10 different runs when evaluating the performance of the balanced and unbalanced setting for cryptocurrency networks. The average results when evaluating the performance on these datasets are reported in Table 5.6.

			***Balanced** Setting*						***Unbalanced** Setting*					
		Algorithm	Precision	Recall	F_1	Accuracy	AU-ROC	AU-PR	Precision	Recall	F_1	Accuracy	AU-ROC	AU-PR
Bitcoin	*RF*	Weber et al. [45]	0.004	0.013	0.009	0.008	0.003	0.002	0.009	0.011	0.007	0.004	0.005	0.007
		node2vec	0.092	0.070	0.082	0.048	0.059	0.066	0.274	0.052	0.087	0.011	0.051	0.087
		RiWalk	0.042	0.088	0.070	0.018	0.028	0.033	0.245	0.037	0.064	0.010	0.050	0.066
	LR	Weber et al. [45]	0.007	0.011	0.006	0.006	0.003	0.004	0.014	0.018	0.014	0.007	0.004	0.020
		node2vec	0.052	0.071	0.081	0.027	0.030	0.038	0.242	0.043	0.073	0.011	0.037	0.059
		RiWalk	0.041	0.084	0.097	0.026	0.021	0.030	0.247	0.035	0.061	0.009	0.042	0.059
	MLP	Weber et al. [45]	0.018	0.020	0.005	0.005	0.004	0.004	0.024	0.018	0.008	0.008	0.004	0.013
		node2vec	0.113	0.256	0.155	0.028	0.030	0.041	0.045	0.079	0.035	0.052	0.042	0.059
		RiWalk	0.093	0.074	0.068	0.017	0.024	0.033	0.048	0.033	0.044	0.043	0.045	0.060
Ethereum	*RF*	trans2vec	0.010	0.007	0.007	0.009	0.008	0.009	0.191	0.003	0.007	0.004	0.013	0.015
		node2vec	0.015	0.009	0.011	0.012	0.008	0.009	0.191	0.003	0.007	0.004	0.013	0.015
		RiWalk	0.011	0.018	0.013	0.013	0.014	0.013	0.300	0.001	0.002	0.004	0.021	0.032
	LR	trans2vec	0.010	0.018	0.008	0.008	0.008	0.009	0.033	0.017	0.023	0.005	0.017	0.015
		node2vec	0.012	0.022	0.013	0.014	0.009	0.010	0.039	0.018	0.022	0.007	0.023	0.017
		RiWalk	0.011	0.032	0.018	0.017	0.013	0.017	0.108	0.005	0.009	0.008	0.023	0.013
	MLP	trans2vec	0.011	0.007	0.008	0.009	0.008	0.010	0.018	0.092	0.025	0.083	0.021	0.021
		node2vec	0.013	0.008	0.010	0.011	0.009	0.009	0.014	0.035	0.021	0.064	0.026	0.030
		RiWalk	0.022	0.038	0.027	0.028	0.026	0.016	0.017	0.067	0.027	0.047	0.043	0.019

Table 5.11: The standard deviation of 10 different runs when evaluating the performance of the balanced and unbalanced setting for cryptocurrency rating networks. The average results when evaluating the performance on these datasets are reported in Table 5.7.

			***Balanced** Setting*						***Unbalanced** Setting*					
		Algorithm	Precision	Recall	F_1	Accuracy	AU-ROC	AU-PR	Precision	Recall	F_1	Accuracy	AU-ROC	AU-PR
Alpha	*RF*	REV2	0.021	0.032	0.067	0.056	0.073	0.099	0.043	0.033	0.058	0.010	0.065	0.022
		node2vec	0.102	0.120	0.091	0.081	0.099	0.109	0.000	0.000	0.000	0.002	0.049	0.005
		RiWalk	0.099	0.095	0.077	0.070	0.070	0.109	0.000	0.000	0.000	0.001	0.067	0.015
	LR	REV2	0.087	0.049	0.071	0.039	0.037	0.019	0.021	0.056	0.009	0.012	0.028	0.017
		node2vec	0.143	0.087	0.088	0.081	0.076	0.108	0.000	0.000	0.000	0.001	0.049	0.025
		RiWalk	0.094	0.111	0.074	0.060	0.065	0.085	0.000	0.000	0.000	0.001	0.054	0.022
	MLP	REV2	0.089	0.093	0.021	0.013	0.077	0.079	0.026	0.019	0.023	0.064	0.042	0.006
		node2vec	0.098	0.149	0.058	0.059	0.104	0.122	0.005	0.131	0.009	0.122	0.055	0.006
		RiWalk	0.139	0.127	0.063	0.071	0.096	0.116	0.031	0.252	0.028	0.232	0.042	0.007
OTC	*RF*	REV2	0.037	0.056	0.044	0.051	0.053	0.047	0.090	0.015	0.023	0.011	0.016	0.022
		node2vec	0.046	0.071	0.047	0.049	0.045	0.048	0.490	0.020	0.038	0.001	0.026	0.028
		RiWalk	0.051	0.091	0.065	0.067	0.061	0.056	0.400	0.016	0.031	0.001	0.043	0.019
	LR	REV2	0.047	0.071	0.048	0.061	0.039	0.047	0.093	0.010	0.010	0.011	0.018	0.022
		node2vec	0.063	0.093	0.064	0.061	0.049	0.050	0.300	0.009	0.017	0.001	0.029	0.034
		RiWalk	0.062	0.099	0.073	0.070	0.060	0.057	0.000	0.000	0.000	0.001	0.027	0.036
	MLP	REV2	0.081	0.073	0.061	0.039	0.089	0.076	0.013	0.038	0.029	0.024	0.037	0.016
		node2vec	0.073	0.135	0.050	0.046	0.101	0.101	0.007	0.073	0.013	0.025	0.037	0.007
		RiWalk	0.103	0.218	0.138	0.078	0.105	0.060	0.006	0.062	0.009	0.032	0.041	0.010

Table 5.12: The standard deviation of 10 different runs when evaluating the performance of the balanced and unbalanced setting for rating networks. The average results when evaluating the performance on these datasets are reported in Table 5.8. **NC* denotes that the node representation approach did not converged in reasonable time (we set the time limit as two days). Thus, we were not able to gain the node embeddings for the classification task.

			***Balanced** Setting*						***Unbalanced** Setting*					
	Algorithm		Precision	Recall	F_1	Accuracy	AU-ROC	AU-PR	Precision	Recall	F_1	Accuracy	AU-ROC	AU-PR
Amazon	*RF*	REV2	0.047	0.088	0.071	0.038	0.039	0.047	0.029	0.027	0.025	0.013	0.025	0.031
		node2vec	0.040	0.098	0.065	0.044	0.058	0.037	0.000	0.000	0.000	0.000	0.040	0.000
		RiWalk	0.045	0.055	0.037	0.043	0.054	0.067	0.000	0.000	0.000	0.000	0.038	0.000
	LR	REV2	0.021	0.031	0.025	0.024	0.022	0.017	0.015	0.026	0.019	0.018	0.022	0.028
		node2vec	0.024	0.000	0.021	0.024	0.000	0.012	0.000	0.000	0.000	0.000	0.000	0.000
		RiWalk	0.024	0.000	0.021	0.024	0.000	0.012	0.000	0.000	0.000	0.000	0.000	0.000
	MLP	REV2	0.079	0.067	0.048	0.049	0.053	0.052	0.029	0.081	0.045	0.098	0.023	0.021
		node2vec	0.211	0.303	0.222	0.016	0.047	0.055	0.000	0.309	0.000	0.310	0.052	0.000
		RiWalk	0.089	0.297	0.145	0.022	0.054	0.052	0.000	0.197	0.000	0.198	0.030	0.000
Epinions	*RF*	REV2	0.024	0.036	0.037	0.021	0.017	0.057	0.054	0.056	0.054	0.013	0.018	0.035
		node2vec	NC*	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC
		RiWalk	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC
	LR	REV2	0.035	0.061	0.032	0.054	0.024	0.043	0.052	0.042	0.064	0.062	0.041	0.075
		node2vec	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC
		RiWalk	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC
	MLP	REV2	0.001	0.032	0.025	0.018	0.005	0.081	0.053	0.073	0.012	0.012	0.018	0.019
		node2vec	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC
		RiWalk	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC

Chapter 6

Towards Better Evaluation for Dynamic Link Prediction

In this chapter, we discuss the extension of our research into the investigation of another important related task on temporal dynamic graphs which is the dynamic link prediction. Here, we discuss how we can investigate the dynamics of a temporal network and how to examine the consistency of relations among networks entities. We also investigate the evaluation setup of the dynamic link prediction task and propose a simple baseline as well as two negative sampling strategies for this task.

6.1 Introduction

Many evolving real-world relations can be modelled by a dynamic graph where nodes correspond to entities and edges represent relations between nodes. Understanding and analyzing the temporal patterns of a dynamic graph is an important open problem. Nodes, edges, weights or attributes can be added, deleted or adjusted over time. For instance, in popular online social networks, many users join the platform on a daily basis while connections between users are constantly added or removed [201]. To facilitate better learning on dynamic graphs, increasing efforts have been devoted to the development of dynamic graph representation learning methods [220, 221, 222, 258, 259].

In particular, the link prediction task focuses on predicting future connections between nodes. Recent methods such as [218, 219, 220, 221, 222] show promising performance on this task with the state-of-the-art (SOTA) performance [221, 222] being often perfect or

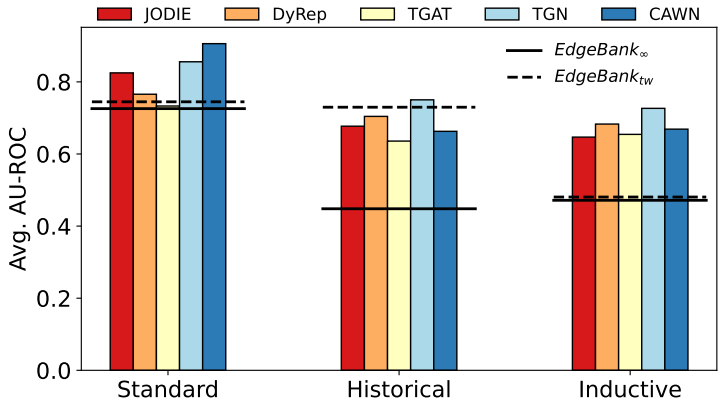


Figure 6.1: The ranking of different methods changes in the proposed negative sampling settings, which eliminate easy negatives. Our proposed baselines (horizontal lines) show competitive performance.

close to perfect on many existing benchmark datasets. However, considering that link prediction in static graphs, an arguably less complex task, still faces major challenges and remains an open problem [188, 260], it is important to examine the near-perfect performance of dynamic link prediction methods. We hypothesize that current evaluation procedures and datasets fail to properly differentiate between proposed approaches.

In this study, we identify drawbacks in the existing evaluation pipeline for dynamic link prediction and propose novel strategies for more robust and effective evaluation. We start by examining the existing benchmark datasets and observe that these datasets are mostly social networks in nature and thus limited in diversity. Therefore, we incorporate 5 new datasets for dynamic link prediction from politics, economics, and (air) transportation. Next, we propose novel visualization techniques for dynamic graphs. We show that in most networks, a significant portion of edges reoccur over time, but the recurrence patterns vary widely across different networks and domains. Based on these observations, we introduce two novel Negative Sampling (NS) strategies, specifically designed for dynamic graphs, which select negative edges based on the recurrence of observed edges. As shown in Fig. 6.1, SOTA methods have a significant decrease in performance when a different set of negative edges is sampled during test time. In addition, the relative ranking of methods changes across NS settings. Therefore, it is important to evaluate methods on different sets of negative edges.

Finally, we introduce a simple memorization-based baseline named EdgeBank which simply stores previously observed edges in memory, and predicts the set of edges in memory as positive at test time. In Fig. 6.1, we contrast the performance of SOTAs

with that of EdgeBank (in horizontal lines). EdgeBank is a surprisingly strong baseline and in the historical NS setting, EdgeBank achieves the second best ranking amongst all methods. As EdgeBank requires neither learning nor hyper-parameter tuning, it is a strong baseline for future methods to compare against.

The goal of this study is to propose more effective evaluation strategies to better differentiate dynamic link prediction methods. Our main contributions can be summarized as follows:

- We identify challenges and drawbacks in the current evaluation of the link prediction task for dynamic graphs: existing strategies for sampling negative edges during evaluation are insufficient, memorization leads to over-optimistic evaluation, and there is a lack of diversity in graph domains.
- We propose a novel non-parameterized and memorization-based method, EdgeBank, as a strong baseline for current and future approaches to compare against.
- We collect and process 5 novel dynamic graph datasets from various domains such as political network, flight network and economics network. These datasets exhibit different temporal edge evolution patterns and can facilitate more robust evaluation.
- Lastly, to evaluate the impact of negative edges on the performance, we outline two novel sampling strategy: *historical NS* and *inductive NS* for selecting negative edges based on the recurrence of previously observed edges in the train and test sets respectively.

6.2 Background

Dynamic graphs can be broadly categorized into *Discrete Time Dynamic Graphs (DTDG)* or *Continuous Time Dynamic Graphs (CTDG)*. While DTDGs comprise a series of static graph snapshots obtained at specific times, CTDGs are more general and have exact temporal information. In this study, we focus on CTDGs, which can be represented by a series of timestamped events $\mathcal{G} = \{\eta(t_1), \eta(t_2), ...\}$ where the timestamps are ordered $(0 \leq t_1 \leq t_2 \leq ...)$. An event $\eta(t)$ can be of different kinds such as edge insertion, edge deletion, node insertion, node deletion, feature alteration, etc.

Dynamic Link Prediction. We investigate the task of predicting the probability of existence of an edge between a node pair at a given timestamp in the future. We divide

edges of a dynamic graphs into three categories: (a) edges that are only seen during training, (b) edges that are seen during training and reappear during test (we denote them as *transductive* edges), and (c) edges that have not been seen during training and only appear during test (we denote them as *inductive* edges).

In our evaluation, we focus on models that are based on Dynamic Graph Neural Networks (DGNNs) which encode structural patterns and aggregate the neighboring node features by a Graph Neural Network (GNN) architecture, while positional encoding or time series models (such as recurrent neural networks (RNNs)) are utilized for encoding temporal patterns. Some of these DGNN models also rely on time embedding methods for capturing inter-event times.

6.3 Dynamic Graph Datasets

We aim to understand the differences between dynamic graph datasets across a variety of domains. To this end, we investigate 7 widely used benchmark datasets and contribute 5 novel dynamic graphs (marked as *new*) from diverse domains currently under-studied in dynamic link prediction literature. The statistics of these datasets are summarized in Table 6.1, and details are explained below:

- **Wikipedia** [218]: consists of edits on Wikipedia pages over one month. The network models editors and the Wiki pages as nodes, and the timestamped posting requests as edges. Edge features are LIWC-feature vectors of edit text.
- **Reddit** [218]: includes the network of one month posts made by users on subreddits, where the nodes specify users or posts and the edges specify the timestamped posting requests.
- **MOOC** [218]: is a network of students interacting with online course content units such as problem sets and videos. Each edge represents a student accessing a specific content unit.
- **LastFM** [218]: is a network of users and songs as nodes where each edge represents a user-listens-to-song relation. The dataset consists of the relations of 1000 users listening to the 1000 most listened songs over a period of one month, and the network is non-attributed.

- **Enron** [261]: is an email correspondence dataset containing around 50K emails exchanged among employees of the ENRON energy company over a three-year period.
- **Social Evolution** [262]: is a mobile phone proximity network which tracks the everyday life of a whole undergraduate dormitory from October 2008 to May 2009.
- **UCI** [263]: is a Facebook-like online communication network among students of the University of California at Irvine, along with timestamps with the temporal granularity of seconds.
- **Flights** (*new*) [264]: is a directed dynamic flight network illustrating the development of air traffic during the COVID-19 pandemic. It was extracted and cleaned for the purpose of this study. Each node represents an airport and each edge is a tracked flight.
- **Canadian Parliament** (*new*) [265]: is a dynamic political network documenting the interactions between Canadian Members of Parliaments (MPs) from 2006 to 2019. Each node is one MP representing an electoral district and each edge is formed when two MPs both voted *"yes"* on a bill.
- **US Legislative** (*new*) [265, 266]: is a senate co-sponsorship network which documents social interactions between legislators from the US Senate. If two politicians co-sponsor a bill, an edge is formed.
- **UN Trade** (*new*) [267]: is a weighted, directed, food and agriculture trading network between 181 nations and spanning over 30 years. The data was originally collected by the Food and Agriculture Organization (FAO) of the United Nations. The weight of each edge is the sum of all trading goods from one nation to another in a year. We extracted and cleaned this dataset for the purpose of this study.
- **UN Vote** (*new*) [268]: is a dataset of roll-call votes in the United Nations General Assembly from 1946 to 2020. If two nations both voted *"yes"* for an item, then the edge weight between them is incremented by one. We extracted and cleaned this dataset for this study.

Table 6.1: Dynamic network dataset statistics.

Measurement	Existing Datasets							**New** Datasets				
	Wikipedia	Reddit	MOOC	LastFM	Enron	Social Evo.	UCI	Flights	Can. Parl.	US Legis.	UN Trade	UN Vote
Domain	Social	Social	Social	Social	Social	Social	Social	Transport	Politics	Politics	Economics	Politics
Nodes	9,227	10,984	7,144	1,980	184	74	1,899	13,169	734	225	255	201
Total edges	157,474	672,447	411,749	1,293,103	125,235	2,099,519	59,835	1,927,145	74,478	60,396	507,497	1,035,742
Uniq. edges	18,257	78,516	178,443	154,993	3,125	4,486	20,296	395,072	51,331	26,423	36,182	31,516
Uniq. timestamps	152,757	669,065	345,600	1,283,614	22,632	565,932	58,911	122	14	12	32	72
Duration	1 month	1 month	17 month	1 month	3 years	8 months	196 days	4 months	14 years	12 congresses	32 years	72 years

6.4 Visualizing Dynamic Graphs

We visualize the differences across dynamic graphs using two types of plots, explained in the following.

6.4.1 Temporal Edge Appearance (TEA) Plot

This illustrates the portion of repeated edges versus newly observed edges for each timestamp in a dynamic graph, as shown in Fig. 6.2. The grey bar indicates the number of edges which were observed in previous time steps and the red bar represents the number of new edges seen at each step. This visualization shows high variance across datasets in temporal evolutionary patterns in terms of new and repeated edges. Some datasets such as Social Evo. comprise mainly repeated edges, while others such as MOOC have a high proportion of new edges. These differences can be important when designing and choosing methods for the link prediction task, because when many edges are repeated, a simple memorization approach can potentially achieve strong performance. On the other hand, if there are many new edges, memorization cannot be sufficient. The TEA plots also show significant differences in when edges occur, and distinctions between our new datasets and existing ones. For example, our new Flights dataset has significantly more unique edges and higher numbers of edges per timestamp.

While the TEA plot shows how many edges are repeated or new overall, it does not directly show how consistent the repeats are. Thus, we next propose:

6.4.2 Temporal Edge Traffic (TET) Plot

This visualizes the recurrence pattern of edges in different dynamic networks over time, as shown in Fig. 6.3. To construct these plots, we first sort edges based on the timestamp they first appear. Then for edges occurring in the same timestamp, we sort them based on when they last occur. Further, we color edges based on whether they are seen in train

only (green), test only (inductive edges, red), or both (transductive edges, orange).

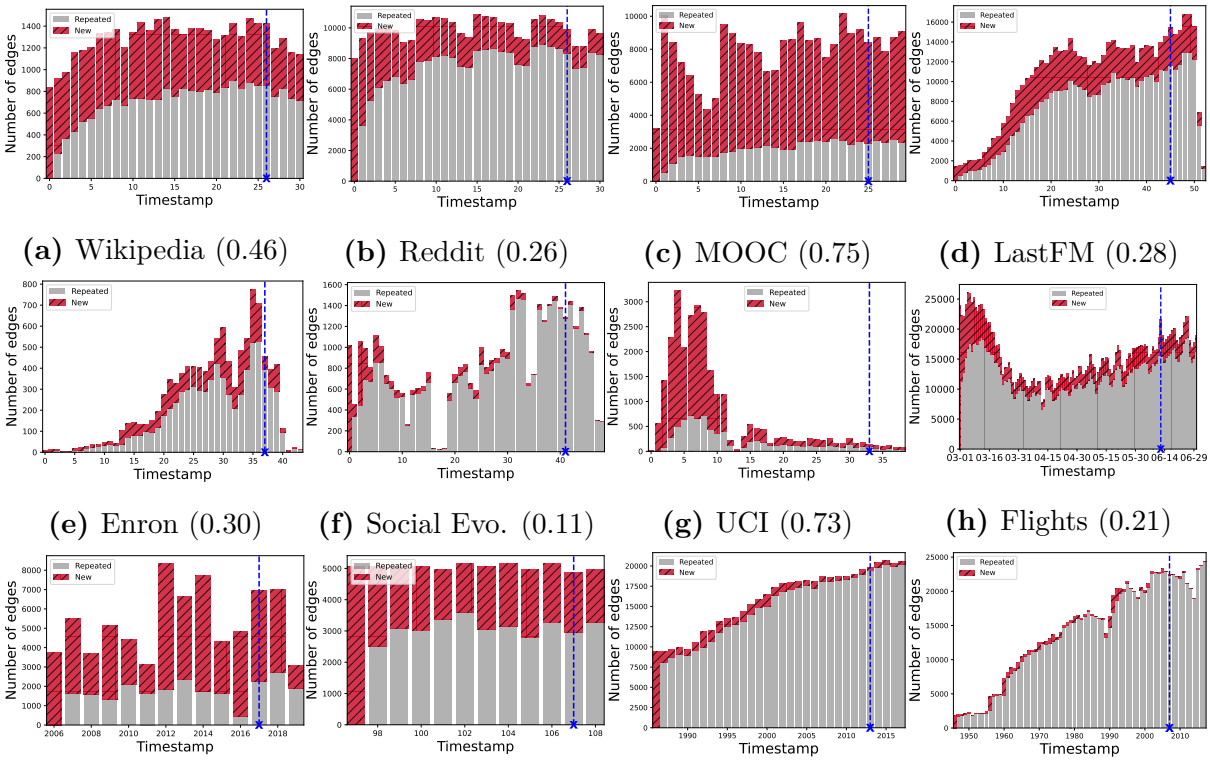
TET plots help us get more insights about the edges that are used for training and testing of different DGNN methods. A memorization approach can potentially predict the transductive positive edges, since it has observed and hence recorded them during training. In particular, if they appear consistently, then simple memorization is likely to be successful. On the other hand, if they appear at some time(s) but then disappear later, then memory is likely still helpful, but simple and full memorization will not work. It would incorrectly predict that those edges still exist. Meanwhile, memorization is not helpful at all for predicting inductive positive test edges at their first appearance, since these are new edges that have not been observed before. For example, while Social Evo. and UN Trade have a relatively similar proportion of repeated vs. new edges based on their TEA plots, we see in their TET plots that UN Trade has far more consistent recurrence. The clear difference we can observe in the visualization is mirrored in the results - the best model on UN Trade is among the worst on Social Evo., and vice versa (Fig. 6.5).

We encourage researchers to investigate the proposed TET plots to get a more comprehensive overview of dynamic graphs in addition to the network statistics.

6.5 EdgeBank Baseline for Dynamic Link Prediction

Observing that many edges in dynamic networks reoccur over time, we want to understand if a simple approach purely based on memorizing past edges can be a competitive baseline. To this end, we propose a pure memorization-based approach called EdgeBank. The memory component of EdgeBank is simply a dictionary which is updated with observed edges at each timestamp. In this way, EdgeBank resembles a *bank* of observed edges and requires no parameters. The storage requirement of EdgeBank is the same as the number of edges in the dataset.

At test time, EdgeBank predicts a test edge as *positive* if the edge was seen before (in the memory), and *negative* otherwise. At each timestamp, EdgeBank updates its memory with newly observed edges, similar to the memory update procedure of TGN [221]. EdgeBank can predict correctly for edges which reoccur frequently over time. There are two scenarios where EdgeBank will make an incorrect prediction: (i) an unseen edge, or (ii) an edge observed before (in memory) that is a negative edge at the



(a) Wikipedia (0.46) **(b)** Reddit (0.26) **(c)** MOOC (0.75) **(d)** LastFM (0.28)

(e) Enron (0.30) **(f)** Social Evo. (0.11) **(g)** UCI (0.73) **(h)** Flights (0.21)

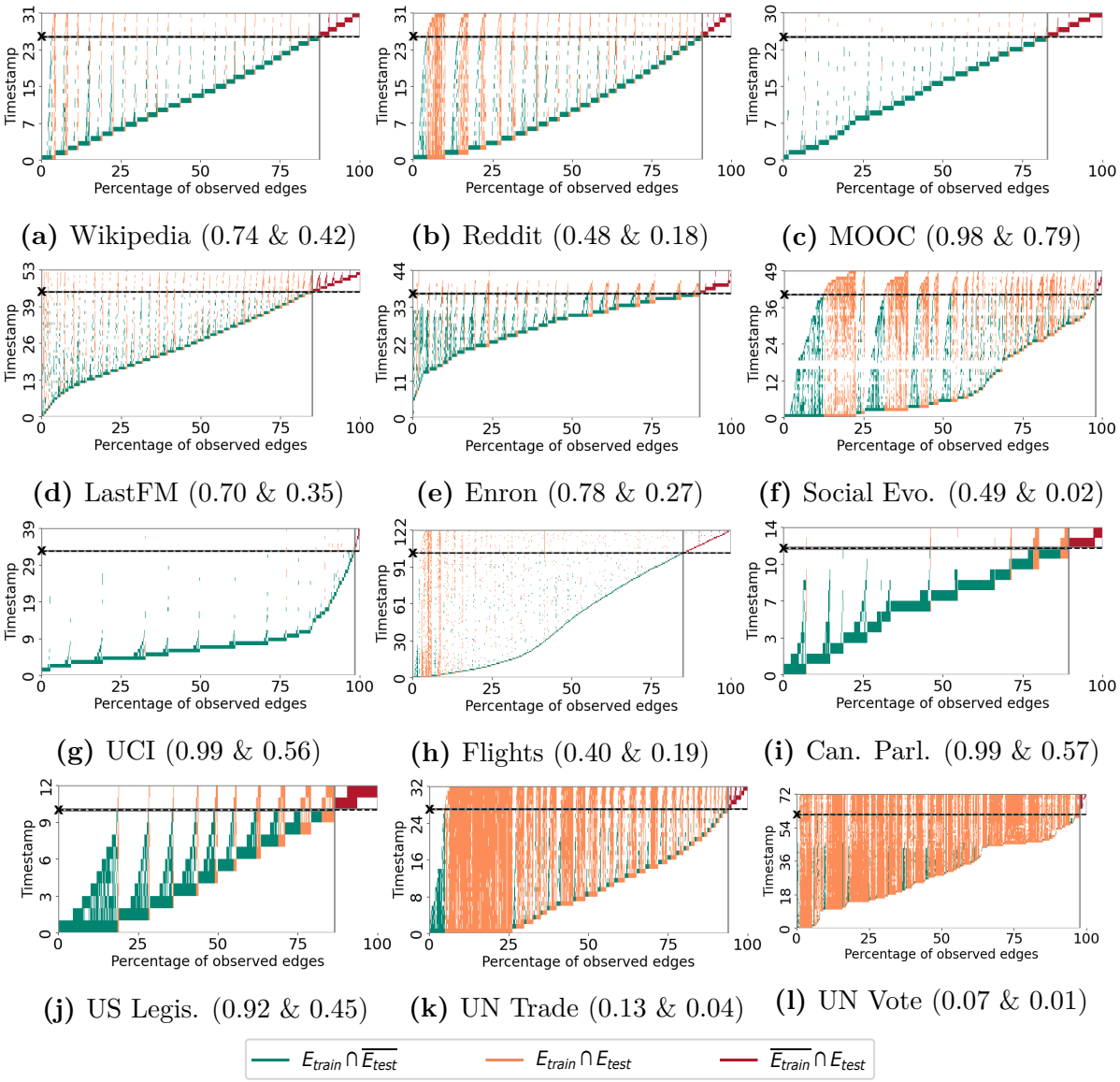
(i) Can. Parl. (0.69) **(j)** US Legis. (0.44) **(k)** UN Trade (0.07) **(l)** UN Vote (0.03)

Figure 6.2: TEA plots show many real world dynamic networks contain a large proportion of edges that reoccur over time. Thus, even a simple memorization approach such as EdgeBank can potentially achieve strong performance. The numbers in parentheses denote the average of the ratio of new to total edges in different timestamps.

current time. In the standard random negative sampling evaluation [220, 221, 222], as graphs are often sparse, it is unlikely that an edge observed before will be sampled as a negative edge. Therefore, EdgeBank has strong performance on negative edges in many cases.

We consider two different memory update strategies for EdgeBank thus resulting in two variants of EdgeBank:

- **EdgeBank$_\infty$**: stores all observed edges in memory, thus remembering edges even from a long time ago. EdgeBank$_\infty$ is prone to false positives on edges which appear once but rarely reoccur over time.
- **EdgeBank$_{\text{tw}}$**: only remembers edges from a fixed sized time window from the immediate past. The size of the time window is set to the duration of test split,



(a) Wikipedia (0.74 & 0.42) **(b)** Reddit (0.48 & 0.18) **(c)** MOOC (0.98 & 0.79)

(d) LastFM (0.70 & 0.35) **(e)** Enron (0.78 & 0.27) **(f)** Social Evo. (0.49 & 0.02)

(g) UCI (0.99 & 0.56) **(h)** Flights (0.40 & 0.19) **(i)** Can. Parl. (0.99 & 0.57)

(j) US Legis. (0.92 & 0.45) **(k)** UN Trade (0.13 & 0.04) **(l)** UN Vote (0.07 & 0.01)

Figure 6.3: TET plots illustrates varied edge traffic patterns in different temporal graphs. Networks are chronologically split into the train and test set, which is the common practice. The horizontal line starting with "**x**" shows the timestamp of the resultant test split. The numbers in parentheses denote $|E_{train} \cap \overline{E_{test}}|/|E_{train}|$ and $|\overline{E_{train}} \cap E_{test}|/|E_{test}|$, respectively.

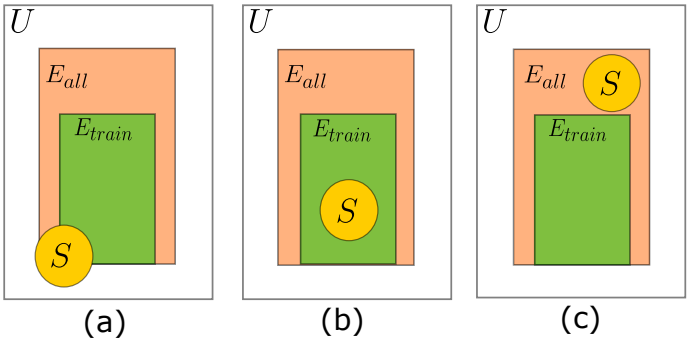


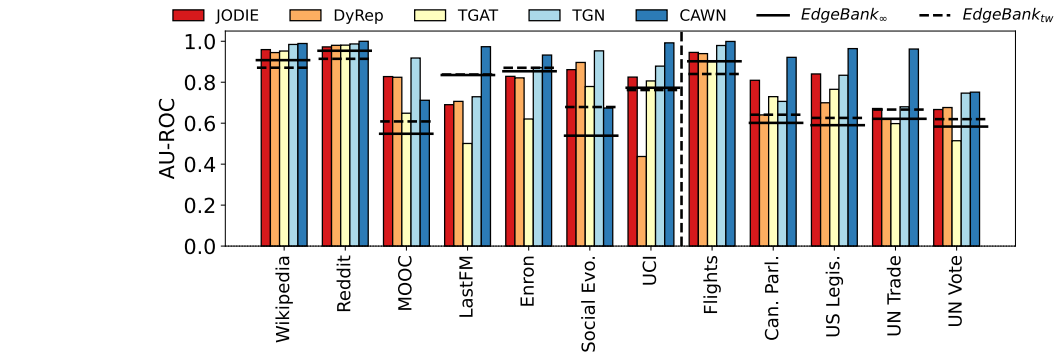
Figure 6.4: Negative edge sampling strategies during evaluation for dynamic link prediction. (a) random sampling (standard in existing work), (b) historical sampling (ours), (c) inductive sampling (ours).

based on the intuition of predicting the test set behavior from the most similar (recent) period in the train set. Thus, $\text{EdgeBank}_{\text{tw}}$ focuses on the edges observed in the short-term past.

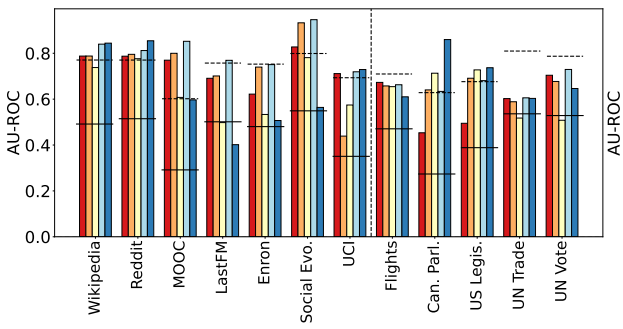
Note that EdgeBank is not designed to replace state-of-the-art methods. Rather we argue that all dynamic graph representation methods should be able to do better than memorization, thus beating EdgeBank. EdgeBank provides a simple and strong baseline to demonstrate how far pure memorization can go on each dataset.

6.6 Negative Sampling in Dynamic Graphs

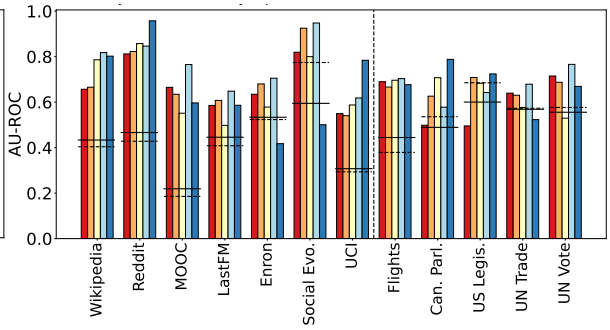
Current SOTA methods for dynamic link prediction often achieve very high performance on existing benchmark datasets [218, 219, 220, 221, 222, 259]. Consequently, one can argue that either the existing datasets are too simplistic or the current evaluation process is insufficient to differentiate methods. We discussed the dataset aspect extensively. Next, we also need to carefully examine the current evaluation setting of DGNNs. In particular, although negative edges constitute half of the evaluation edges, little attention has been dedicated to understanding the effect of different sets of negative edges on the overall performance. In this section, we take a closer look at Negative Sampling (NS) strategies for evaluation of dynamic link prediction, and propose two novel NS strategies for more robust evaluation and better differentiation amongst methods. To better motivate the two new methods, we first explain the standard NS strategy widely used in the literature.



(a) Standard setting.



(b) Historical setting.



(c) Inductive setting.

Figure 6.5: Performance of different methods in three evaluation settings based on the negative sampling approach.

6.6.1 Random Negative Sampling

Current evaluation samples negative edges randomly from almost all possible node pairs of the networks [218, 219, 220, 221, 222]. At each time step, we have a set of positive edges consisting of source and destination nodes together with edge timestamps and edge features. To generate negative samples, the standard procedure is to keep the timestamps, features, and source nodes of the positive edges, while choosing destination nodes randomly from all nodes. This has two significant issues:

(1) No Collision Checking: existing implementations have no collision check between positive and negative edges. Therefore, it is possible for the same edge to be both positive and negative. This collision is more likely to happen in denser datasets, such as UN Vote and UN Trade. A basic accept-reject sampling could address this issue, as applied in our experiments.

(2) No Reoccurring Edges: the probability of sampling an edge which was observed before is often very low due to the sparsity of the graph. Therefore, a simple method

like EdgeBank can perform well on negative edges. However, in many real-world tasks such as flight prediction, correct prediction of the same edge for different time steps is particularly important. For example, predicting that, yet again, there will be no flight between the north and south poles this week is not nearly as practical as predicting whether a standard commuter flight will be canceled.

To address this second issue, we need to sample from previously observed edges, which can be from train or test set. This constitutes the two alternative NS strategies proposed here, illustrated in Fig. 6.4. Here S is the sample space for negative edges. Let U, E_{all}, E_{train} be the set of all possible node pairs, all edges in the dataset (train and test) and all edges in the train set, respectively. Note that $E_{all} = E_{train} + E_{test}$ where E_{test} is all edges in the test set. Lastly, we set $U_{neg} = U - E_{all}$. Now, in random NS, we sample from edges $e \in U$, with the proportion from E_{all} and E_{train} regulated only by the sizes of those sets relative to U. To resolve the issues with random NS, in the following sections we propose *historical NS* and *inductive NS*.

6.6.2 Historical Negative Sampling

In historical NS, we focus on sampling negative edges from the set of edges that have been observed during previous timestamps but are absent in the current step. The objective of this strategy is to evaluate if a given method is able to predict in which timestamps an edge would reoccur, rather than, for example, naively predicting it always reoccurs whenever it has been seen once. Therefore, in *historical NS*, for a given time step t, we sample from the edges $e \in (E_{train} \cap \overline{E_t})$. It should be noted that if the number of available historical edges is insufficient to match the number of positive edges, the remaining negative edges are sampled by the random NS strategy.

6.6.3 Inductive Negative Sampling

While in *historical NS* we focus on observed edges from the training set, in *inductive NS*, our focus is to evaluate whether a given method can model the recurrence pattern of edges only seen during test time. At test time, after edges unseen during training are observed, the model is asked to predict if such edges exist in future steps of the test phase. Therefore, in *inductive NS*, we sample from the edges $e \in (E_{test} \cap \overline{E_{train}} \cap \overline{E_t})$ at time step t. As these edges are not observed during training, they are considered as *inductive* edges. Similar to before, if the number of inductive negative edges is not adequate, the remaining negative edges are sampled by the random NS strategy.

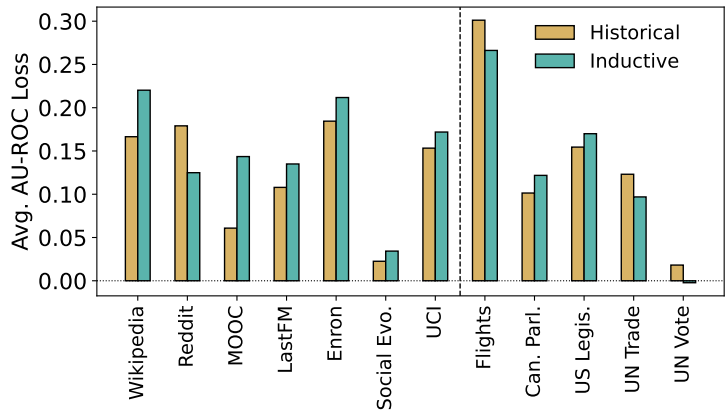
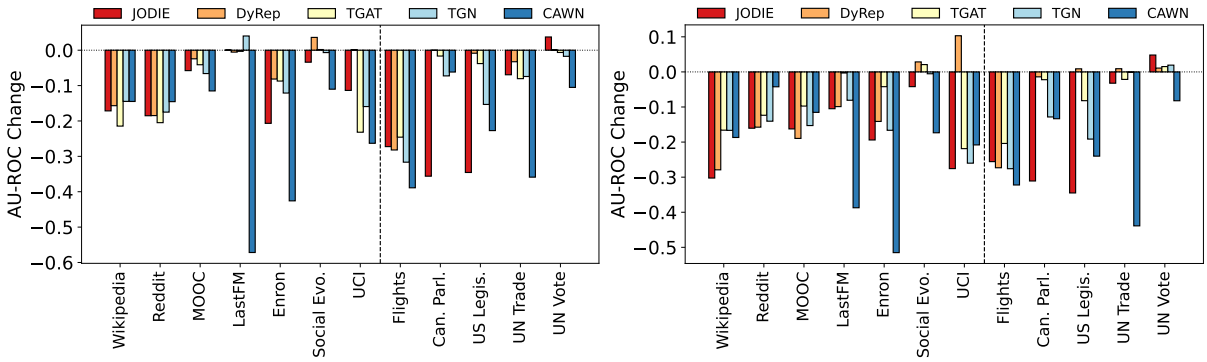


Figure 6.6: Average AU-ROC loss (lower = better) of SOTA methods with different NSs. The impact of moving to historical or inductive NS varies across datasets.

6.7 Experiments



(a) Performance change in historical compared to standard NS.

(b) Performance change in inductive compared to standard NS.

Figure 6.7: Analyzing performance change (higher = better) of SOTAs in the historical and inductive settings.

In this section, we present a comprehensive evaluation of the dynamic link prediction task on all 12 datasets with 5 SOTA methods. Our experimental setup closely follows [218, 219, 220, 221, 222]. The objective of the link prediction task is to predict the existence of an edge at a given time between a node pair. For all DGNN based methods, we use a Multilayer Perceptron as the final output layer for edge prediction, where the concatenated node embeddings are the input and the probability of the edge is the output. For all experiments, we use the same $70\% - 15\% - 15\%$ chronological splits for the train-

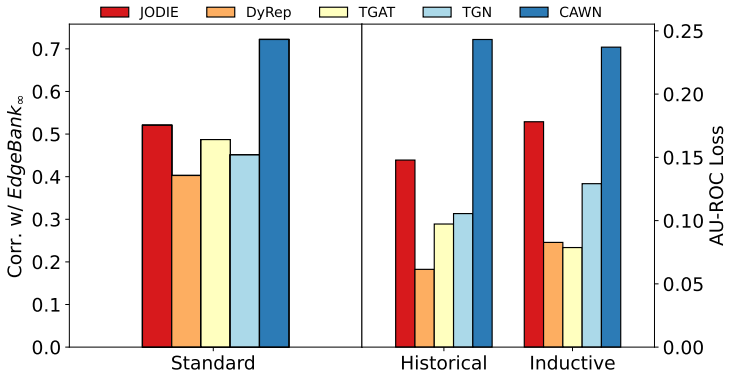


Figure 6.8: Performance correlation with the proposed memorization baseline, $EdgeBank_{\infty}$ (on the left), predicts the performance loss (lower = better) of the methods in both of the harder negative sampling settings (on the right).

validation-test sets as [220, 221, 222]. The averaged results over five runs are reported. The *Area Under Receiver Operating Characteristic (AU-ROC)* metric is selected as the main performance metric. We visualize the results for easier interpretation, but the exact numbers that produce the visualizations – as well as the equivalents with *Average Precision (AP)* – are presented in the appendix.

Fig. 6.5a compares the performance of all models under the standard random negative sampling strategy. First, we observe significant variation in performance for all models across datasets. This supports the benefits of evaluation on datasets from different domains. Second, we observe a strong inconsistency in relative ranking amongst methods across datasets. For example, while CAWN achieves SOTA on most datasets, on MOOC and Social Evo. it performs significantly worse than several other models. Lastly, note that EdgeBank demonstrates competitive performance even when compared against SOTA methods. Despite being a simple baseline, EdgeBank outperforms highly parametrized and complex models on some datasets such as LastFM, Enron and UN Trade.

Next, we examine the impact of negative sampling strategies on performance. Fig. 6.5b and Fig. 6.5c shows the performance of different methods with the *historical NS* and *inductive NS* strategies, respectively. First, we observe that the ranking of models can change significantly across different NS settings. This shows that relying on a single NS strategy such as the random NS is insufficient for the complete evaluation of methods. Second, for the *historical NS* setting, $\text{EdgeBank}_{\text{tw}}$ becomes highly competitive, often beating most methods and even achieving SOTA for UN Trade, UN

Vote, Flights and Enron. This shows that in these datasets, recently observed edges contain crucial information for link prediction. Third, EdgeBank$_\infty$ has a significant drop in performance in both NS strategies. This shows that as the negative edges are sampled from either previously observed edges or unseen edges, naively memorizing all past edges is no longer sufficient. However, EdgeBank can perform competitively under random NS. This further shows that the standard random NS is limited in its ability to effectively differentiate methods.

Fig. 6.6 shows the average drop in performance with historical and inductive NS across different SOTAs. In general, the decrease is at least 10 percentage points. The new Flights dataset is particularly challenging, with nearly 30 percentage points average loss when comparing historical NS with random NS. This means models struggle to correctly predict whether a flight that happened in the past will happen again.

In Fig. 6.7, we examine the performance drops for each model in the historical or inductive NS setting. CAWN, which performed best overall with random NS, collapses on certain datasets such as LastFM and Enron. Other models fare much better on these datasets. All models exhibit a large performance drop on the Flights dataset.

The performance degradation is also correlated with the degree of memorization. Fig. 6.8 shows that the models which are more correlated with EdgeBank$_\infty$ tend to perform worse in the historical and inductive NS settings. Since EdgeBank$_\infty$ is naively dependent on the memory, higher correlation with it indicates a model relies more heavily on memorization. For example, CAWN has the highest correlation and JODIE the second highest. They have the largest and second largest losses (respectively) in performance with the more challenging negative sampling. Similarly, DyRep is the least correlated with EdgeBank, and experiences the least drop in performance with historical NS and second least with inductive NS.

6.8 Conclusion

In this study, we have presented four tools to improve evaluation of models for dynamic link prediction. First, we created new visualizations (TEA and especially TET plots) to better understand the patterns of temporal edges in different dynamic network datasets. Second, we introduced five new datasets which provide new diversity and challenges for modeling. Third, we showed limitations of random negative sampling and introduced two new strategies for negative sampling (historical and inductive) to overcome these limitations and more thoroughly evaluate models. Lastly, we proposed a simple but

competitive baseline, EdgeBank. It can also yield insights into how much different models rely on memorization, in addition to helping understand to what degree memorization is effective on a specific dataset.

Thorough evaluation is critical to producing better models. When we applied these tools to compare existing models, we found that performance and ranking of different models varies significantly. We hope that these tools will provide practical ways to improve the evaluation and overcome the limitations of the current standard process, and in turn help produce models that are rigorously superior to existing ones.

In future work, we aim to synthesize insights here into concise measurements that summarize the difficulty of different datasets. We will also expand this work to temporal node classification, another key task. Finally, we hope to apply our tools to produce novel and effective DGNN models.

6.9 Appendix: Extended Results

Here, we report the extended results used to plot the figures in main body of Chapter 6.

Table 6.2: Average precision, standard setting with **random** negative sampling.

Method	Wikipedia	Reddit	MOOC	LastFM	Enron	Social Evo.	UCI	Flights	Can. Parl.	US Legis.	UN Trade	UN Vote
JODIE	0.95	0.95	0.78	0.68	0.78	0.79	0.75	0.94	0.75	0.76	0.64	0.64
DyRep	0.95	0.98	0.80	0.71	0.80	0.87	0.46	0.93	0.58	0.64	0.61	0.65
TGAT	0.95	0.98	0.61	0.50	0.59	0.76	0.78	0.89	0.68	0.70	0.58	0.52
TGN	0.99	0.99	0.90	0.72	0.85	0.93	0.88	0.98	0.64	0.77	0.64	0.71
CAWN	0.99	0.99	0.75	0.98	0.95	0.72	0.99	0.99	0.94	0.97	0.97	0.82
$\text{EdgeBank}_{\text{tw}}$	0.87	0.91	0.58	0.79	0.84	0.61	0.76	0.84	0.65	0.58	0.60	0.57
EdgeBank_{∞}	0.90	0.95	0.53	0.77	0.80	0.52	0.76	0.89	0.60	0.55	0.57	0.55

Table 6.3: AU-ROC, standard setting with **random** negative sampling.

Method	Wikipedia	Reddit	MOOC	LastFM	Enron	Social Evo.	UCI	Flights	Can. Parl.	US Legis.	UN Trade	UN Vote
JODIE	0.96	0.97	0.83	0.69	0.83	0.86	0.83	0.95	0.81	0.84	0.67	0.67
DyRep	0.94	0.98	0.82	0.71	0.82	0.90	0.44	0.94	0.64	0.70	0.62	0.68
TGAT	0.95	0.98	0.65	0.50	0.62	0.78	0.81	0.90	0.73	0.77	0.60	0.51
TGN	0.98	0.99	0.91	0.73	0.87	0.95	0.88	0.80	0.71	0.83	0.68	0.75
CAWN	0.99	0.99	0.71	0.97	0.93	0.67	0.99	0.99	0.92	0.96	0.96	0.75
$\text{EdgeBank}_{\text{tw}}$	0.87	0.91	0.61	0.84	0.87	0.68	0.76	0.84	0.64	0.63	0.67	0.62
EdgeBank_{∞}	0.91	0.95	0.55	0.84	0.85	0.54	0.77	0.90	0.60	0.59	0.62	0.58

Table 6.4: Average precision, *historical* negative sampling. The number in the parentheses show the performance loss compared to the standard setting. The intensity of the color relates to the amount of loss.

Method	Wikipedia	Reddit	MOOC	LastFM	Enron	Social Evo.	UCI	Flights	Can. Parl.	US Legis.	UN Trade	UN Vote
JODIE	0.77 (0.18)	0.77 (0.18)	0.70 (0.08)	0.68 (0.00)	0.56 (0.22)	0.73 (0.07)	0.62 (0.13)	0.65 (0.29)	0.43 (0.31)	0.45 (0.31)	0.56 (0.08)	0.66 (-0.02)
DyRep	0.81 (0.14)	0.79 (0.19)	0.74 (0.06)	0.71 (0.00)	0.71 (0.08)	0.93 (-0.06)	0.45 (0.01)	0.63 (0.30)	0.57 (0.01)	0.63 (0.02)	0.58 (0.03)	0.64 (0.01)
TGAT	0.76 (0.19)	0.77 (0.21)	0.59 (0.02)	0.50 (0.00)	0.53 (0.06)	0.77 (-0.01)	0.61 (0.18)	0.65 (0.24)	0.67 (0.02)	0.63 (0.07)	0.51 (0.07)	0.51 (0.01)
TGN	0.88 (0.11)	0.81 (0.18)	0.84 (0.06)	0.76 (-0.05)	0.72 (0.13)	0.95 (-0.01)	0.76 (0.12)	0.64 (0.34)	0.56 (0.08)	0.56 (0.21)	0.57 (0.07)	0.67 (0.04)
CAWN	0.89 (0.10)	0.89 (0.11)	0.66 (0.09)	0.56 (0.42)	0.63 (0.32)	0.64 (0.08)	0.79 (0.20)	0.63 (0.37)	0.90 (0.05)	0.82 (0.16)	0.72 (0.25)	0.75 (0.07)
EdgeBank$_{tw}$	0.71 (0.16)	0.70 (0.21)	0.57 (0.01)	0.69 (0.10)	0.68 (0.15)	0.71 (-0.10)	0.65 (0.10)	0.65 (0.18)	0.64 (0.01)	0.63 (-0.05)	0.73 (-0.13)	0.71 (-0.13)
EdgeBank$_{\infty}$	0.50 (0.41)	0.51 (0.44)	0.43 (0.10)	0.50 (0.27)	0.50 (0.31)	0.53 (-0.01)	0.44 (0.32)	0.49 (0.41)	0.48 (0.12)	0.46 (0.10)	0.52 (0.05)	0.51 (0.03)

Table 6.5: AU-ROC, *historical* negative sampling. The number in the parentheses show the performance loss compared to the standard setting. The intensity of the color relates to the amount of loss.

Method	Wikipedia	Reddit	MOOC	LastFM	Enron	Social Evo.	UCI	Flights	Can. Parl.	US Legis.	UN Trade	UN Vote
JODIE	0.79 (0.17)	0.79 (0.19)	0.77 (0.06)	0.69 (0.00)	0.62 (0.21)	0.83 (0.03)	0.71 (0.11)	0.67 (0.27)	0.45 (0.36)	0.49 (0.35)	0.60 (0.07)	0.70 (-0.04)
DyRep	0.79 (0.16)	0.80 (0.18)	0.80 (0.02)	0.70 (0.01)	0.74 (0.08)	0.93 (-0.04)	0.44 (0.00)	0.66 (0.28)	0.64 (0.00)	0.69 (0.01)	0.59 (0.03)	0.68 (0.00)
TGAT	0.74 (0.21)	0.78 (0.20)	0.61 (0.04)	0.50 (0.00)	0.53 (0.09)	0.78 (0.00)	0.57 (0.23)	0.65 (0.25)	0.71 (0.02)	0.73 (0.04)	0.52 (0.08)	0.51 (0.01)
TGN	0.84 (0.14)	0.81 (0.18)	0.85 (0.07)	0.77 (-0.04)	0.75 (0.12)	0.95 (0.01)	0.72 (0.16)	0.66 (0.32)	0.63 (0.07)	0.68 (0.15)	0.61 (0.07)	0.73 (0.02)
CAWN	0.84 (0.14)	0.85 (0.15)	0.60 (0.12)	0.40 (0.57)	0.51 (0.43)	0.56 (0.11)	0.73 (0.26)	0.61 (0.39)	0.86 (0.06)	0.74 (0.23)	0.60 (0.36)	0.65 (0.11)
EdgeBank$_{tw}$	0.77 (0.10)	0.77 (0.14)	0.60 (0.01)	0.76 (0.08)	0.75 (0.12)	0.80 (-0.12)	0.69 (0.07)	0.71 (0.13)	0.63 (0.01)	0.68 (-0.05)	0.81 (-0.14)	0.79 (-0.17)
EdgeBank$_{\infty}$	0.49 (0.42)	0.51 (0.44)	0.29 (0.26)	0.50 (0.33)	0.48 (0.37)	0.55 (-0.01)	0.35 (0.42)	0.47 (0.43)	0.27 (0.33)	0.39 (0.20)	0.54 (0.09)	0.53 (0.05)

Table 6.6: Average precision, *inductive* negative sampling. The number in the parentheses show the performance loss compared to the standard setting. The intensity of the color relates to the amount of loss.

Method	Wikipedia	Reddit	MOOC	LastFM	Enron	Social Evo.	UCI	Flights	Can. Parl.	US Legis.	UN Trade	UN Vote
JODIE	0.66 (0.28)	0.84 (0.11)	0.66 (0.12)	0.60 (0.08)	0.59 (0.18)	0.72 (0.08)	0.49 (0.26)	0.68 (0.25)	0.48 (0.27)	0.45 (0.31)	0.59 (0.05)	0.67 (-0.02)
DyRep	0.69 (0.26)	0.85 (0.13)	0.63 (0.17)	0.63 (0.08)	0.67 (0.12)	0.92 (-0.05)	0.54 (-0.08)	0.65 (0.28)	0.57 (0.01)	0.64 (0.00)	0.62 (-0.01)	0.64 (0.01)
TGAT	0.82 (0.14)	0.88 (0.10)	0.54 (0.06)	0.50 (0.00)	0.57 (0.02)	0.79 (-0.03)	0.62 (0.17)	0.70 (0.19)	0.67 (0.01)	0.58 (0.12)	0.54 (0.04)	0.51 (0.00)
TGN	0.87 (0.12)	0.88 (0.11)	0.77 (0.13)	0.67 (0.04)	0.70 (0.15)	0.95 (-0.01)	0.69 (0.19)	0.69 (0.28)	0.52 (0.12)	0.53 (0.24)	0.63 (0.01)	0.70 (0.01)
CAWN	0.86 (0.13)	0.97 (0.03)	0.66 (0.09)	0.70 (0.28)	0.57 (0.38)	0.60 (0.12)	0.83 (0.16)	0.69 (0.30)	0.85 (0.10)	0.81 (0.17)	0.67 (0.31)	0.76 (0.06)
EdgeBank$_{tw}$	0.46 (0.41)	0.47 (0.44)	0.42 (0.16)	0.46 (0.33)	0.54 (0.30)	0.69 (-0.08)	0.43 (0.32)	0.47 (0.37)	0.59 (0.05)	0.65 (-0.06)	0.56 (0.04)	0.55 (0.03)
EdgeBank$_{\infty}$	0.48 (0.43)	0.49 (0.46)	0.42 (0.11)	0.48 (0.30)	0.54 (0.26)	0.55 (-0.03)	0.44 (0.33)	0.49 (0.40)	0.55 (0.06)	0.56 (-0.01)	0.55 (0.02)	0.53 (0.02)

Table 6.7: AU-ROC, *inductive* negative sampling. The number in the parentheses show the performance loss compared to the standard setting. The intensity of the color relates to the amount of loss.

Method	Wikipedia	Reddit	MOOC	LastFM	Enron	Social Evo.	UCI	Flights	Can. Parl.	US Legis.	UN Trade	UN Vote
JODIE	0.66 (0.30)	0.81 (0.16)	0.67 (0.16)	0.59 (0.10)	0.63 (0.19)	0.82 (0.04)	0.55 (0.28)	0.69 (0.26)	0.50 (0.31)	0.50 (0.35)	0.64 (0.03)	0.71 (-0.05)
DyRep	0.67 (0.28)	0.82 (0.16)	0.63 (0.19)	0.61 (0.10)	0.68 (0.14)	0.93 (-0.03)	0.54 (-0.10)	0.67 (0.27)	0.63 (0.01)	0.71 (-0.01)	0.63 (-0.01)	0.69 (-0.01)
TGAT	0.79 (0.17)	0.86 (0.12)	0.55 (0.10)	0.50 (0.00)	0.58 (0.04)	0.80 (-0.02)	0.59 (0.22)	0.70 (0.20)	0.71 (0.02)	0.68 (0.08)	0.58 (0.02)	0.53 (-0.02)
TGN	0.82 (0.17)	0.85 (0.14)	0.77 (0.15)	0.65 (0.08)	0.71 (0.17)	0.95 (0.01)	0.62 (0.26)	0.70 (0.28)	0.58 (0.13)	0.64 (0.19)	0.68 (0.00)	0.77 (-0.02)
CAWN	0.80 (0.19)	0.96 (0.04)	0.60 (0.12)	0.59 (0.39)	0.42 (0.52)	0.50 (0.17)	0.78 (0.21)	0.68 (0.32)	0.79 (0.13)	0.72 (0.24)	0.52 (0.44)	0.67 (0.08)
EdgeBank$_{tw}$	0.40 (0.47)	0.43 (0.49)	0.19 (0.42)	0.41 (0.43)	0.52 (0.35)	0.77 (-0.10)	0.29 (0.47)	0.38 (0.46)	0.54 (0.11)	0.69 (-0.06)	0.57 (0.09)	0.58 (0.04)
EdgeBank$_{\infty}$	0.43 (0.47)	0.47 (0.49)	0.22 (0.33)	0.45 (0.39)	0.53 (0.32)	0.59 (-0.06)	0.31 (0.47)	0.44 (0.46)	0.49 (0.11)	0.60 (-0.01)	0.57 (0.05)	0.56 (0.03)

Chapter 7

Conclusion

In this dissertation, we started by considering the task of anomaly detection in cryptocurrency networks. Public availability of the transaction histories of cryptocurrency networks provides abundant opportunities for investigation of human behaviour in financial domains. Nevertheless, leveraging this massive, high-dimensional, large-scale data is not an easy task. An imperative step prior to designing new analytical solutions is to design data features or representations that are proportionate with the complexity of the relationships in cryptocurrency networks. It is essential to have representations that can accommodate the high-dimensional, multi-modal, and relational nature of the data in order to truly leverage the potential of the network data. In this dissertation, we proposed efficient techniques for generating features aiming to detect anomalous entities in cryptocurrency networks. These features incorporated domain knowledge extracted from the investigation of the cryptocurrency networks as well as more general attributes obtained through complex network analysis. Further, we considered the generalization of the proposed features to encompass a more general model of temporal weighted graphs, and demonstrated that the proposed anomaly detection approach achieved state-of-the-art results in domains beyond cryptocurrency networks such as rating and social networks. Meanwhile, we examined the challenges of graph anomaly detection task utilizing node classification techniques, which is commonplace in the literature. We demonstrated the significance of performance measures and evaluation setups on the actual performance of different methods. Ultimately, we concentrated on one of the important research topic on networks, i.e. learning on dynamic graphs, that has diverse applications such as social networks, recommender systems, drug discovery and many more. We precisely inspected the evaluation setting of dynamic link prediction tasks and provided more

robust approaches for the evaluation of dynamic graph learning methods.

At the core of this dissertation is the notion of graph anomaly detection, fundamentals of blockchain-based cryptocurrency networks, and dynamic networks. We presented a comprehensive overview of these quickly-expanding areas of research (Chapter 2). Focusing on cryptocurrency transactions, we first proposed methods for detecting malicious accounts on the Ethereum network through generating a limited set of features being leveraged by machine learning algorithms (Chapter 3). Then, we extended the feature set with useful additional features exploiting the relational characteristics of the data, and prolong our anomaly detection scheme to handle cryptocurrency networks with diverse architectural models (Chapter 4). We further expanded our research in two more directions. First, we generalized the anomaly detection task to other kinds of real-world networks, such as rating platforms and social networks, that could demonstrate dissimilar characteristics from the already investigated cryptocurrency networks. Second, we investigated scenarios where the node labels vary over time, and introduced solutions to reflect the dynamic evolution of the networks in feature sets. Additionally, we contrasted dynamic and static node classification (Chapter 5). We also closely inspected the importance of the evaluation setting as well as performance metrics when leveraging node classification for anomaly detection purposes. Finally, centring our attention on dynamic networks and the quickly expanding trend of research in this nascent area, we introduced the challenges of current evaluation settings for the dynamic link prediction task, and presented new datasets, novel visualization techniques, as well as evaluation strategies to further facilitate the development of dynamic graph representation learning methods (Chapter 6).

7.1 Future Directions

While we explored several important directions for anomaly detection in cryptocurrency and other real-world networks, many questions remain unanswered and multiple directions are open to explore.

7.1.1 Anomaly Detection on Dynamic Graphs

Dynamic graph representation learning is still in its infancy phase and many important tasks, such as anomaly detection, are almost untouched when it comes to dynamic networks. While anomaly detection involves several challenges including scarcity of the

labelled data, unbalanced datasets, and difficulty in extracting real data due to security issues, learning tasks on dynamic graphs exacerbates the difficulties. Analysis of the dynamic graphs requires an understanding of graph-structured relations as well as dynamic evolution of the networks. Additionally, the anomalous patterns can appear as structural or temporal anomalies. The increased difficulty of anomaly detection on dynamic graphs requires significant works to provide efficient methods for the full realization of this task.

7.1.2 Learning on Dynamic Networks with Unbalanced Labels

Although there are several approaches for resolving the dataset imbalance issue, it is relatively unexplored in the realm of dynamic networks. In many different applications, such as anomaly detection, fraud detection, intrusion detection, misinformation detection, combating human trafficking, etc., the underlying data not only contains relational information, but also demonstrates considerable unbalanced patterns among instances of different classes. Particularly on dynamic graphs, the scarcity of the appropriate labels can appear in different forms. For instance, the records of network activities at specific timestamps can be inadequate, we may be restricted to access specific parts of the networks which results in a scarcity of the structural information about those regions of the networks, or different types of nodes or edges of the networks may not necessarily have similar distributions regarding the label dispensations. Therefore, it is of great importance to be cognizant of the imbalance issue of the real-world networks and design methods that meet the requirements for tackling this mostly overlooked issue.

7.1.3 Efficient Incorporation of Interpretable Features in GNNs

Improving the interpretability of representation learning methods, specially GNNs, is one of the most pressing challenges. One important factor that may impede the widespread adoption of representation learning methods in areas other than computer science is the uninterpretability of these methods. In contrast, feature engineering approaches provide more insight into the analysis of network characteristics. We showed that our elaborated set of features is efficient in detecting anomalies in instances of real-world applications in diverse domains and settings. An interesting future direction is to efficiently fuse these features in GNN architectures. This objective has two forthright advantages. First, incorporating the well-defined features in GNNs could help

in increasing the interpretability of representation learning methods. Second, incorporating automatic representation learning methods helps relieve the painstaking and brittle feature engineering process.

7.1.4 Consistent Benchmarks and Evaluation Setups

Like other areas of science, one important driver of progress in dynamic graph representation learning is the availability of benchmark tasks and datasets. Currently, the available dynamic network datasets come from domains that are not much diverse. Moreover, these datasets are rarely labelled and, in most cases, demonstrate highly skewed labels. Also, these datasets are not interesting, realistic, or original enough to maintain the surging interests of the research community. Therefore, it is critical to enforce this challenging area of research by providing rich, diverse, and appropriate datasets. Moreover, there is a considerable lack of a consistent and efficient evaluation setup for assessing the dynamic graph learning methods. There is no consistent terminology, performance metrics, and evaluation setup that are efficient enough to differentiate the true performance of different learning methods. The development of the benchmark datasets and consistent evaluation setup are critical factors in future growth of this area, since they will make a common ground for comparing various methods and help further advance the reproducibility culture of algorithmic progress.

Chapter 8

List of Publications

1. F. Poursafaei, A. Huang, K. Perline, and R. Rabbany, "Towards Better Evaluation for Dynamic Link Prediction." arXiv preprint arXiv:2207.10128. 2022.

2. F. Poursafaei, Z. Zilic, and R. Rabbany, "A Strong Node Classification Baseline for Temporal Graphs." in *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*. SIAM, 2022.

3. F. Poursafaei, Z. Zilic, and R. Rabbany, "On Anomaly Detection in Graphs as Node Classification," in *2022 the 4th International Conference on Big Data Engineering and Technology (BDET)*, 2022.

4. F. Poursafaei, R. Rabbany, and Z. Zilic, "Sigtran: Signature Vectors for Detecting Illicit Activities in Blockchain Transaction Networks." in *Proceedings of the 25th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-2021)*. Springer, 2021, pp. 27–39.

5. F. Poursafaei, G. B. Hamad, and Z. Zilic, "Detecting Malicious Ethereum Entities via Application of Machine Learning Classification," in *2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*. IEEE, 2020, pp. 120–127.

Bibliography

[1] T. McGhin, K.-K. R. Choo, C. Z. Liu, and D. He, "Blockchain in Healthcare Applications: Research Challenges and Opportunities," *Journal of Network and Computer Applications*, vol. 135, pp. 62–75, 2019.

[2] A. Hasselgren, K. Kralevska, D. Gligoroski, S. A. Pedersen, and A. Faxvaag, "Blockchain in Healthcare and Health Sciences: A Scoping Review," *International Journal of Medical Informatics*, vol. 134, p. 104040, 2020.

[3] T. M. Fernández-Caramés and P. Fraga-Lamas, "A Review on the Use of Blockchain for the Internet of Things," *IEEE Access*, vol. 6, pp. 32 979–33 001, 2018.

[4] H.-N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for Internet of Things: A Survey," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8076–8094, 2019.

[5] "CoinMarketCap," https://coinmarketcap.com/currencies/ethereum/, accessed: 2022-07-17.

[6] "Statista," https://www.statista.com/statistics/863917/number-crypto-coins-tokens/, (Accessed on 04/10/2022).

[7] K. Grauer, W. Kushner, and H. Updegrave, "Original Data and Research into Cryptocurrency-based Crime," Chainalysis Inc., The 2022 Crypto Crime Report, 2022.

[8] J. Li, C. Gu, F. Wei, and X. Chen, "A Survey on Blockchain Anomaly Detection Using Data Mining Techniques," in *International Conference on Blockchain and Trustworthy Systems*. Springer, 2019, pp. 491–504.

[9] M. Conti, E. S. Kumar, C. Lal, and S. Ruj, "A Survey on Security and Privacy Issues of Bitcoin," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3416–3452, 2018.

[10] X. F. Liu, X.-J. Jiang, S.-H. Liu, and C. K. Tse, "Knowledge Discovery in Cryptocurrency Transactions: A Survey," *IEEE Access*, vol. 9, pp. 37 229–37 254, 2021.

[11] V. Buterin *et al.*, "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform," *White Paper*, vol. 3, no. 37, pp. 2–1, 2014.

[12] S. Ferretti and G. D'Angelo, "On the Ethereum Blockchain Structure: A Complex Networks Theory Perspective," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 12, p. e5493, 2020.

[13] M. U. Hassan, M. H. Rehmani, and J. Chen, "Anomaly Detection in Blockchain Networks: A Comprehensive Survey," *arXiv preprint arXiv:2112.06089*, 2021.

[14] M. Rahouti, K. Xiong, and N. Ghani, "Bitcoin Concepts, Threats, and Machine-Learning Security Solutions," *IEEE Access*, vol. 6, pp. 67 189–67 205, 2018.

[15] S. Rouhani and R. Deters, "Security, Performance, and Applications of Smart Contracts: A Systematic Survey," *IEEE Access*, vol. 7, pp. 50 759–50 779, 2019.

[16] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu, "A Comprehensive Survey on Graph Anomaly Detection with Deep Learning," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[17] W. Meng, E. W. Tischhauser, Q. Wang, Y. Wang, and J. Han, "When Intrusion Detection Meets Blockchain Technology: A Review," *IEEE Access*, vol. 6, pp. 10 179–10 188, 2018.

[18] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[19] F. Poursafaei, G. B. Hamad, and Z. Zilic, "Detecting Malicious Ethereum Entities via Application of Machine Learning Classification," in *2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*. IEEE, 2020, pp. 120–127.

[20] F. Poursafaei, R. Rabbany, and Z. Zilic, "SigTran: Signature Vectors for Detecting Illicit Activities in Blockchain Transaction Networks," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. Springer, 2021, pp. 27–39.

[21] F. Poursafaei, Z. Zilic, and R. Rabbany, "A Strong Node Classification Baseline for Temporal Graphs," in *Proceedings of the SIAM International Conference on Data Mining (SDM)*. SIAM, 2022.

[22] ——, "On Anomaly Detection in Graphs as Node Classification," in *The 4th International Conference on Big Data Engineering and Technology (BDET)*, 2022.

[23] H. Chen, M. Pendleton, L. Njilla, and S. Xu, "A Survey on Ethereum Systems Security: Vulnerabilities, Attacks, and Defenses," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–43, 2020.

[24] S. Dey, "Securing Majority-Attack In Blockchain Using Machine Learning And Algorithmic Game Theory: A Proof of Work," *arXiv preprint arXiv:1806.05477*, 2018.

[25] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making Smart Contracts Smarter," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 254–269.

[26] J. Garay, A. Kiayias, and N. Leonardos, "The Bitcoin Backbone Protocol: Analysis and Applications," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 281–310.

[27] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," *White Paper*, 2008.

[28] M. Bartoletti, B. Pes, and S. Serusi, "Data Mining for Detecting Bitcoin Ponzi Schemes," in *Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE, 2018, pp. 75–84.

[29] F. Reid and M. Harrigan, "An Analysis of Anonymity in the Bitcoin System," in *Security and Privacy in Social Networks*. Springer, 2013, pp. 197–223.

[30] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A Fistful of Bitcoins: Characterizing Payments Among Men with No Names," in *Proceedings of the Conference on Internet Measurement Conference*. ACM, 2013, pp. 127–140.

[31] M. Möser and R. Böhme, "Anonymous Alone? Measuring Bitcoin's Second-Generation Anonymization Techniques," in *IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2017, pp. 32–41.

[32] M. Spagnuolo, F. Maggi, and S. Zanero, "Bitiodine: Extracting Intelligence from the Bitcoin Network," in *International Conference on Financial Cryptography and Data Security*. Springer, 2014, pp. 457–468.

[33] J. H. Ziegeldorf, R. Matzutt, M. Henze, F. Grossmann, and K. Wehrle, "Secure and Anonymous Decentralized Bitcoin Mixing," *Future Generation Computer Systems*, vol. 80, pp. 448–466, 2018.

[34] M. Möser and R. Böhme, "The Price of Anonymity: Empirical Evidence from a Market for Bitcoin Anonymization," *Journal of Cybersecurity*, vol. 3, no. 2, pp. 127–135, 2017.

[35] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, "Mixcoin: Anonymity for Bitcoin with Accountable Mixes," in *International Conference on Financial Cryptography and Data Security*. Springer, 2014, pp. 486–504.

[36] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating User Privacy in Bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2013, pp. 34–51.

[37] T. Slattery, "Taking a Bit Out of Crime: Bitcoin and Cross-Border Tax Evasion," *Brook. J. Int'l L.*, vol. 39, p. 829, 2014.

[38] M. Möser, R. Böhme, and D. Breuker, "Towards Risk Scoring of Bitcoin Transactions," in *International Conference on Financial Cryptography and Data Security*. Springer, 2014, pp. 16–32.

[39] T. Moore, "The Promise and Perils of Digital Currencies," 2013.

[40] M. Vasek and T. Moore, "Analyzing the Bitcoin Ponzi Scheme Ecosystem," in *International Conference on Financial Cryptography and Data Security*. Springer, 2018, pp. 101–112.

[41] ——, "There'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*. Springer, 2015, pp. 44–61.

[42] M. Möser, R. Böhme, and D. Breuker, "An Inquiry into Money Laundering Tools in the Bitcoin Ecosystem," in *2013 APWG e-Crime Researchers Summit*. IEEE, 2013, pp. 1–14.

[43] C. Brenig, G. Müller *et al.*, "Economic Analysis of Cryptocurrency Backed Money Laundering," 2015.

[44] S. Jovicic and Q. Tan, "Machine Learning for Money Laundering Detection in the Blockchain Financial Transaction System," *Journal of Fundamental and Applied Sciences*, vol. 10, no. 4S, pp. 376–381, 2018.

[45] M. Weber, G. Domeniconi, J. Chen, D. K. I. Weidele, C. Bellei, T. Robinson, and C. E. Leiserson, "Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics," *arXiv preprint arXiv:1908.02591*, 2019.

[46] A. H. H. Kabla, M. Anbar, S. Manickam, T. A. Alamiedy, P. B. Cruspe, A. K. Al-Ani, and S. Karupayah, "Applicability of Intrusion Detection System on Ethereum Attacks: A Comprehensive Review," *IEEE Access*, 2022.

[47] "Ethereum Statistics," https://www.alchemy.com/overviews/ethereum-statistics, accessed: 2022-07-17.

[48] H. Zhou, A. Milani Fard, and A. Makanju, "The State of Ethereum Smart Contracts Security: Vulnerabilities, Countermeasures, and Tool Support," *Journal of Cybersecurity and Privacy*, vol. 2, no. 2, pp. 358–378, 2022.

[49] O. Sürücü, U. Yeprem, C. Wilkinson, W. Hilal, S. A. Gadsden, J. Yawney, N. Alsadi, and A. Giuliano, "A Survey on Ethereum Smart Contract Vulnerability Detection Using Machine Learning," *Disruptive Technologies in Information Sciences VI*, vol. 12117, pp. 110–121, 2022.

[50] S. S. Kushwaha, S. Joshi, D. Singh, M. Kaur, and H.-N. Lee, "Systematic Review of Security Vulnerabilities in Ethereum Blockchain Smart Contract," *IEEE Access*, 2022.

[51] "The DAO," https://github.com/blockchainsllc/DAO/tree/v1.0, accessed: 2021-12-25.

[52] "The Parity Wallet Hack Explained," https://blog.openzeppelin.com/on-the-parity-wallet-multisig-hack-405a8c12e8f7/, accessed: 2021-12-25.

[53] "EIP-20: Token Standard," https://eips.ethereum.org/EIPS/eip-20, accessed: 2021-12-26.

[54] C. G. Akcora, Y. R. Gel, and M. Kantarcioglu, "Blockchain Networks: Data Structures of Bitcoin, Monero, Zcash, Ethereum, Ripple, and Iota," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 12, no. 1, p. 1436, 2022.

[55] H. Arslanian, "Ethereum," in *The Book of Crypto*. Springer, 2022, pp. 91–98.

[56] P. M. Monamo, V. Marivate, and B. Twala, "A Multifaceted Approach to Bitcoin Fraud Detection: Global and Local Outliers," in *15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2016, pp. 188–194.

[57] T. Pham and S. Lee, "Anomaly Detection in Bitcoin Network Using Unsupervised Learning Methods," *arXiv preprint arXiv:1611.03941*, 2016.

[58] A. Bogner, "Seeing is Understanding: Anomaly Detection in Blockchains with Visualized Features," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the ACM International Symposium on Wearable Computers*. ACM, 2017, pp. 5–8.

[59] P. Monamo, V. Marivate, and B. Twala, "Unsupervised Learning for Robust Bitcoin Fraud Detection," in *Information Security for South Africa (ISSA)*. IEEE, 2016, pp. 129–134.

[60] S. K. Shaukat and V. J. Ribeiro, "RansomWall: A Layered Defense System Against Cryptographic Ransomware Attacks Using Machine Learning," in *10th International Conference on Communication Systems & Networks (COMSNETS)*. IEEE, 2018, pp. 356–363.

[61] W. Chen, Z. Zheng, E. C.-H. Ngai, P. Zheng, and Y. Zhou, "Exploiting Blockchain Data to Detect Smart Ponzi Schemes on Ethereum," *IEEE Access*, vol. 7, pp. 37 575–37 586, 2019.

[62] M. A. Harlev, H. Sun Yin, K. C. Langenheldt, R. Mukkamala, and R. Vatrapu, "Breaking Bad: De-Anonymising Entity Types on the Bitcoin Blockchain Using Supervised Machine Learning," in *Proceedings of the 51st Hawaii International Conference on System Sciences*, 2018.

[63] D. Ermilov, M. Panov, and Y. Yanovich, "Automatic Bitcoin Address Clustering," in *16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2017, pp. 461–466.

[64] J. J. Xu, "Are Blockchains Immune to All Malicious Attacks?" *Financial Innovation*, vol. 2, no. 1, p. 25, 2016.

[65] N. Atzei, M. Bartoletti, and T. Cimoli, "A Survey of Attacks on Ethereum Smart Contracts (sok)," in *Principles of Security and Trust*. Springer, 2017, pp. 164–186.

[66] W. Chen, Z. Zheng, J. Cui, E. Ngai, P. Zheng, and Y. Zhou, "Detecting Ponzi Schemes on Ethereum: Towards Healthier Blockchain Technology," in *Proceedings of the World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 2018, pp. 1409–1418.

[67] J. Hirshman, Y. Huang, and S. Macke, "Unsupervised Approaches to Detecting Anomalous Behavior in the Bitcoin Transaction Network," Technical Report, Stanford University, Tech. Rep., 2013.

[68] J. Payette, S. Schwager, and J. Murphy, "Characterizing the Ethereum Address Space," 2017.

[69] K. Delmolino, M. Arnett, A. Kosba, A. Miller, and E. Shi, "Step by Step Towards Creating a Safe Smart Contract: Lessons and Insights from a Cryptocurrency Lab," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 79–94.

[70] T. H.-D. Huang, "Hunting the Ethereum Smart Contract: Color-Inspired Inspection of Potential Attacks," *arXiv preprint arXiv:1807.01868*, 2018.

[71] M. Di Angelo and G. Salzer, "A Survey of Tools for Analyzing Ethereum Smart Contracts," in *IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*. IEEE, 2019, pp. 69–78.

[72] J. Liu and Z. Liu, "A Survey on Security Verification of Blockchain Smart Contracts," *IEEE Access*, vol. 7, pp. 77 894–77 904, 2019.

[73] G. Destefanis, M. Marchesi, M. Ortu, R. Tonelli, A. Bracciali, and R. Hierons, "Smart Contracts Vulnerabilities: A Call for Blockchain Software Engineering," in *International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*. IEEE, 2018, pp. 19–25.

[74] R. Norvill, B. B. F. Pontiveros, R. State, I. Awan, and A. Cullen, "Automated Labeling of Unknown Contracts in Ethereum," in *26th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2017, pp. 1–6.

[75] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, "Data Mining for Credit Card Fraud: A Comparative Study," *Decision Support Systems*, vol. 50, no. 3, pp. 602–613, 2011.

[76] A. Dal Pozzolo, O. Caelen, Y.-A. Le Borgne, S. Waterschoot, and G. Bontempi, "Learned Lessons in Credit Card Fraud Detection from a Practitioner Perspective," *Expert systems with applications*, vol. 41, no. 10, pp. 4915–4928, 2014.

[77] N. Carneiro, G. Figueira, and M. Costa, "A Data Mining Based System for Credit-Card Fraud Detection in e-Tail," *Decision Support Systems*, vol. 95, pp. 91–101, 2017.

[78] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[79] J. Lorenz, M. I. Silva, D. Aparício, J. T. Ascensão, and P. Bizarro, "Machine Learning Methods to Detect Money Laundering in the Bitcoin Blockchain in the Presence of Label Scarcity," *arXiv preprint arXiv:2005.14635*, 2020.

[80] S. Farrugia, J. Ellul, and G. Azzopardi, "Detection of Illicit Accounts over the Ethereum Blockchain," *Expert Systems with Applications*, vol. 150, p. 113318, 2020.

[81] Z. Yuan, Q. Yuan, and J. Wu, "Phishing Detection on Ethereum via Learning Representation of Transaction Subgraphs," in *International Conference on Blockchain and Trustworthy Systems*. Springer, 2020, pp. 178–191.

[82] T. Pham and S. Lee, "Anomaly Detection in the Bitcoin System: A Network Perspective," *arXiv preprint arXiv:1611.03942*, 2016.

[83] Y. Hu, S. Seneviratne, K. Thilakarathna, K. Fukuda, and A. Seneviratne, "Characterizing and Detecting Money Laundering Activities on the Bitcoin Network," *arXiv preprint arXiv:1912.12060*, 2019.

[84] J. Wu, D. Lin, Z. Zheng, and Q. Yuan, "T-EDGE: Temporal Weighted Multidigraph Embedding for Ethereum Transaction Network Analysis," *arXiv preprint arXiv:1905.08038*, 2019.

[85] J. Wu, Q. Yuan, D. Lin, W. You, W. Chen, C. Chen, and Z. Zheng, "Who Are the Phishers? Phishing Scam Detection on Ethereum via Network Embedding," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

[86] D. Lin, J. Wu, Q. Yuan, and Z. Zheng, "Modeling and Understanding Ethereum Transaction Records via A Complex Network Approach," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2020.

[87] A. Khan, "Graph Analysis of the Ethereum Blockchain Data: A Survey of Datasets, Methods, and Future Work," 2022.

[88] J. Wu, J. Liu, Y. Zhao, and Z. Zheng, "Analysis of Cryptocurrency Transactions from a Network Perspective: An Overview," *Journal of Network and Computer Applications*, vol. 190, p. 103139, 2021.

[89] Y. Xie, J. Jin, J. Zhang, S. Yu, and Q. Xuan, "Temporal-Amount Snapshot MultiGraph for Ethereum Transaction Tracking," in *International Conference on Blockchain and Trustworthy Systems*. Springer, 2021, pp. 133–146.

[90] Y. Xie, J. Zhou, J. Wang, J. Zhang, Y. Sheng, J. Wu, and Q. Xuan, "Understanding Ethereum Transactions via Network Approach," in *Graph Data Mining*. Springer, 2021, pp. 155–176.

[91] J. Liang, L. Li, and D. Zeng, "Evolutionary Dynamics of Cryptocurrency Transaction Networks: An Empirical Study," *PloS one*, vol. 13, no. 8, p. e0202202, 2018.

[92] R. Kher, S. Terjesen, and C. Liu, "Blockchain, Bitcoin, and ICOs: A Review and Research Agenda," *Small Business Economics*, vol. 56, no. 4, pp. 1699–1720, 2021.

[93] A. A. Monrat, O. Schelén, and K. Andersson, "A Survey of Blockchain from the Perspectives of Applications, Challenges, and Opportunities," *IEEE Access*, vol. 7, pp. 117 134–117 151, 2019.

[94] M. Bhowmik, T. S. S. Chandana, and B. Rudra, "Comparative Study of Machine Learning Algorithms for Fraud Detection in Blockchain," in *5th International Conference on Computing Methodologies and Communication (ICCMC)*. IEEE, 2021, pp. 539–541.

[95] L. Chen, J. Peng, Y. Liu, J. Li, F. Xie, and Z. Zheng, "Phishing Scams Detection in Ethereum Transaction Network," *ACM Transactions on Internet Technology (TOIT)*, vol. 21, no. 1, pp. 1–16, 2020.

[96] A. K. Mandal and P. Dinda, "A Survey on Unsupervised Machine Learning Approach for Fraud Detection in Bitcoin," *AIJR Abstracts*, p. 33, 2022.

[97] Z. Shahbazi and Y.-C. Byun, "Knowledge Discovery on Cryptocurrency Exchange Rate Prediction Using Machine Learning Pipelines," *Sensors*, vol. 22, no. 5, p. 1740, 2022.

[98] S. E. Charandabi and K. Kamyar, "Prediction of Cryptocurrency Price Index Using Artificial Neural Networks: A Survey of the Literature," *European Journal of Business and Management Research*, vol. 6, no. 6, pp. 17–20, 2021.

[99] S. A. H. Havidz, V. E. Karman, and I. Y. Mambea, "Is Bitcoin Price Driven by Macro-Financial Factors and Liquidity? A Global Consumer Survey Empirical Study," *Organizations and Markets in Emerging Economies*, vol. 12, no. 2, pp. 399–414, 2021.

[100] A. M. Khedr, I. Arif, M. El-Bannany, S. M. Alhashmi, and M. Sreedharan, "Cryptocurrency Price Prediction Using Traditional Statistical and Machine-Learning Techniques: A Survey," *Intelligent Systems in Accounting, Finance and Management*, vol. 28, no. 1, pp. 3–34, 2021.

[101] R. G. Tiwari, A. K. Agarwal, R. K. Kaushal, and N. Kumar, "Prophetic Analysis of Bitcoin Price Using Machine Learning Approaches," in *6th International Conference on Signal Processing, Computing and Control (ISPCC)*. IEEE, 2021, pp. 428–432.

[102] Y. Mezquita, A. B. Gil-González, J. Prieto, and J. M. Corchado, "Cryptocurrencies and Price Prediction: A Survey," in *International Congress on Blockchain and Applications*. Springer, 2021, pp. 339–346.

[103] M. Iqbal, M. S. Iqbal, F. H. Jaskani, K. Iqbal, and A. Hassan, "Time-Series Prediction of Cryptocurrency Market Using Machine Learning Techniques," *EAI Endorsed Transactions on Creative Technologies*, p. e4, 2021.

[104] F. Ozer and C. O. Sakar, "An Automated Cryptocurrency Trading System Based on the Detection of Unusual Price Movements with a Time-Series Clustering-Based approach," *Expert Systems with Applications*, vol. 200, p. 117017, 2022.

[105] M. J. Hamayel and A. Y. Owda, "A Novel Cryptocurrency Price Prediction Model Using GRU, LSTM and bi-LSTM Machine Learning Algorithms," *AI*, vol. 2, no. 4, pp. 477–496, 2021.

[106] L. Lai, T. Zhou, Z. Cai, Z. Liang, and H. Bai, "A Survey on Security Threats and Solutions of Bitcoin," *Journal of Cybersecurity*, vol. 3, no. 1, p. 29, 2021.

[107] Z. Wang, H. Jin, W. Dai, K.-K. R. Choo, and D. Zou, "Ethereum Smart Contract Security Research: Survey and Future Research Opportunities," *Frontiers of Computer Science*, vol. 15, no. 2, pp. 1–18, 2021.

[108] E. Rabieinejad, A. Yazdinejad, and R. M. Parizi, "A Deep Learning Model for Threat Hunting in Ethereum Blockchain," in *IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2021, pp. 1185–1190.

[109] A. Alqahtani and F. T. Sheldon, "A Survey of Crypto Ransomware Attack Detection Methodologies: An Evolving Outlook," *Sensors*, vol. 22, no. 5, p. 1837, 2022.

[110] S. UmaMaheswaran, D. Uike, K. Ramachandran, A. Tharangini, T. Suba, and D. Verma, "The Critical Understanding on the Emerging Threats and Defensive Aspects in Cryptocurrencies using Machine Learning Techniques," in *2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. IEEE, 2022, pp. 1938–1942.

[111] N. Rani, S. V. Dhavale, A. Singh, and A. Mehra, "A Survey on Machine Learning-Based Ransomware Detection," in *Proceedings of the 7th International Conference on Mathematics and Computing*. Springer, 2022, pp. 171–186.

[112] A. Trozze, J. Kamps, E. A. Akartuna, F. J. Hetzel, B. Kleinberg, T. Davies, and S. D. Johnson, "Cryptocurrencies and Future Financial Crime," *Crime Science*, vol. 11, no. 1, pp. 1–35, 2022.

[113] U. Urooj, B. A. S. Al-rimy, A. Zainal, F. A. Ghaleb, and M. A. Rassam, "Ransomware Detection Using the Dynamic Analysis and Machine Learning: A Survey and Research Directions," *Applied Sciences*, vol. 12, no. 1, p. 172, 2021.

[114] D. Khan, L. T. Jung, and M. A. Hashmani, "Systematic Literature Review of Challenges in Blockchain Scalability," *Applied Sciences*, vol. 11, no. 20, p. 9372, 2021.

[115] M. Lu, Z. Han, S. X. Rao, Z. Zhang, Y. Zhao, Y. Shan, R. Raghunathan, C. Zhang, and J. Jiang, "BRIGHT - Graph Neural Networks in Real-Time Fraud Detection," *arXiv preprint arXiv:2205.13084*, 2022.

[116] G. Zhang, Z. Li, J. Huang, J. Wu, C. Zhou, J. Yang, and J. Gao, "eFraudCom: An e-Commerce Fraud Detection System via Competitive Graph Neural Networks," *ACM Transactions on Information Systems (TOIS)*, vol. 40, no. 3, pp. 1–29, 2022.

[117] Q. Lai, J. Tian, W. Wang, and X. Hu, "Spatial-Temporal Attention Graph Convolution Network on Edge Cloud for Traffic Flow Prediction," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[118] N. Sharma, I. Kaushik, B. Bhushan, C. K. Dixit *et al.*, "Cryptocurrency Revolution: Bitcoin Time Forecasting & Blockchain Anomaly Detection," in *Blockchain Technology in Healthcare Applications*. CRC Press, 2022, pp. 61–85.

[119] K. Martin, M. Rahouti, M. Ayyash, and I. Alsmadi, "Anomaly Detection in Blockchain Using Network Representation and Machine Learning," *Security and Privacy*, vol. 5, no. 2, p. e192, 2022.

[120] S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. Subrahmanian, "REV2: Fraudulent User Prediction in Rating Platforms," in *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, 2018, pp. 333–341.

[121] J. Tang, J. Li, Z. Gao, and J. Li, "Rethinking Graph Neural Networks for Anomaly Detection," *arXiv preprint arXiv:2205.15508*, 2022.

[122] A. J. Minnich, N. Chavoshi, A. Mueen, S. Luan, and M. Faloutsos, "TrueView: Harnessing the Power of Multiple Review Sites," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 787–797.

[123] N. Hussain, H. Turab Mirza, G. Rasool, I. Hussain, and M. Kaleem, "Spam Review Detection Techniques: A Systematic Literature Review," *Applied Sciences*, vol. 9, no. 5, p. 987, 2019.

[124] W. Chen, X. Guo, Z. Chen, Z. Zheng, and Y. Lu, "Phishing Scam Detection on Ethereum: Towards Financial Security for Blockchain Ecosystem," in *International Joint Conferences on Artificial Intelligence Organization*, 2020, pp. 4506–4512.

[125] C. Akcora, "BitcoinHeist: Topological Data Analysis for Ransomware Prediction on the Bitcoin Blockchain," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

[126] P. Nerurkar, S. Bhirud, D. Patel, R. Ludinard, Y. Busnel, and S. Kumari, "Supervised Learning Model for Identifying Illegal Activities in Bitcoin," *Applied Intelligence*, pp. 1–20, 2020.

[127] L. Nan and D. Tao, "Bitcoin Mixing Detection Using Deep Autoencoder," in *IEEE 3rd International Conference on Data Science in Cyberspace (DSC)*. IEEE, 2018, pp. 280–287.

[128] S. Sayadi, S. B. Rejeb, and Z. Choukair, "Anomaly Detection Model over Blockchain Electronic Transactions," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 2019, pp. 895–900.

[129] J. Kim, K. Kim, G.-Y. Jeon, and M. Sohn, "Temporal Patterns Discovery of Evolving Graphs for Graph Neural Network (GNN)-based Anomaly Detection in Heterogeneous Networks," *Journal of Internet Services and Information Security*, vol. 12, no. 1, pp. 72–82, 2022.

[130] O. Atkinson, A. Bhardwaj, C. Englert, V. S. Ngairangbam, and M. Spannowsky, "Anomaly Detection with Convolutional Graph Neural Networks," *Journal of High Energy Physics*, vol. 2021, no. 8, pp. 1–19, 2021.

[131] L. Xie, D. Pi, X. Zhang, J. Chen, Y. Luo, and W. Yu, "Graph Neural Network Approach for Anomaly Detection," *Measurement*, vol. 180, p. 109546, 2021.

[132] M. Jin, Y. Liu, Y. Zheng, L. Chi, Y.-F. Li, and S. Pan, "ANEMONE: Graph Anomaly Detection with Multi-Scale Contrastive Learning," in *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, 2021, pp. 3122–3126.

[133] Y. Liu, X. Ao, Z. Qin, J. Chi, J. Feng, H. Yang, and Q. He, "Pick and Choose: A GNN-Based Imbalanced Learning Approach for Fraud Detection," in *Proceedings of the Web Conference*, 2021, pp. 3168–3177.

[134] A. Goodge, B. Hooi, S.-K. Ng, and W. S. Ng, "LUNAR: Unifying Local Outlier Detection Methods via Graph Neural Networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 6, 2022, pp. 6737–6745.

[135] A. Deng and B. Hooi, "Graph Neural Network-based Anomaly Detection in Multivariate Time Series," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 4027–4035.

[136] R. Longadge and S. Dongre, "Class Imbalance Problem in Data Mining Review," *arXiv preprint arXiv:1305.1707*, 2013.

[137] C. Drumnond, "Class Imbalance and Cost Sensitivity: Why Undersampling Beats Oversampling," in *ICML-KDD Workshop: Learning from Imbalanced Datasets*, vol. 3, 2003.

[138] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[139] Z.-H. Zhou and X.-Y. Liu, "Training Cost-Sensitive Neural Networks with Methods Addressing the Class Imbalance Problem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 63–77, 2005.

[140] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "SMOTEBoost: Improving Prediction of the Minority Class in Boosting," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2003, pp. 107–119.

[141] H. He and E. A. Garcia, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[142] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart, "Representation Learning for Dynamic Graphs: A Survey," *Journal of Machine Learning Research*, vol. 21, no. 70, pp. 1–73, 2020.

[143] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network Representation Learning: A Survey," *IEEE Transactions on Big Data*, vol. 6, no. 1, pp. 3–28, 2018.

[144] J. Jin, M. Heimann, D. Jin, and D. Koutra, "Toward Understanding and Evaluating Structural Node Embeddings," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 16, no. 3, pp. 1–32, 2021.

[145] P. Jiao, X. Guo, T. Pan, W. Zhang, Y. Pei, and L. Pan, "A Survey on Role-Oriented Network Embedding," *IEEE Transactions on Big Data*, 2021.

[146] M. Scholkemper and M. T. Schaub, "Local, Global and Scale-Dependent Node Roles," in *IEEE International Conference on Autonomous Systems (ICAS)*. IEEE, 2021, pp. 1–5.

[147] I. Chami, S. Abu-El-Haija, B. Perozzi, C. Ré, and K. Murphy, "Machine Learning on Graphs: A Model and Comprehensive Taxonomy," *arXiv preprint arXiv:2005.03675*, 2020.

[148] A. Grover and J. Leskovec, "node2vec: Scalable Feature Learning for Networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 855–864.

[149] X. Ma, G. Qin, Z. Qiu, M. Zheng, and Z. Wang, "RiWalk: Fast Structural Node Embedding via Role Identification," in *IEEE International Conference on Data Mining (ICDM)*. IEEE, 2019, pp. 478–487.

[150] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and Their Compositionality," *Advances in Neural Information Processing Systems*, vol. 26, pp. 3111–3119, 2013.

[151] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online Learning of Social Representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.

[152] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[153] P. Li, Y. Wang, H. Wang, and J. Leskovec, "Distance Encoding: Design Provably More Powerful Neural Networks for Graph Representation Learning," *arXiv preprint arXiv:2009.00142*, 2020.

[154] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," *arXiv preprint arXiv:1609.02907*, 2016.

[155] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 1025–1035.

[156] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph Attention Networks," *arXiv preprint arXiv:1710.10903*, 2017.

[157] F. Chen, Y.-C. Wang, B. Wang, and C.-C. J. Kuo, "Graph Representation Learning: A Survey," *APSIPA Transactions on Signal and Information Processing*, vol. 9, 2020.

[158] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How Powerful Are Graph Neural Networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[159] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A Comprehensive Survey on Graph Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2020.

[160] Y. Zhou, H. Zheng, X. Huang, S. Hao, D. Li, and J. Zhao, "Graph Neural Networks: Taxonomy, Advances, and Trends," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 1, pp. 1–54, 2022.

[161] J. M. Thomas, A. Moallemy-Oureh, S. Beddar-Wiesing, and C. Holzhüter, "Graph Neural Networks Designed for Different Graph Types: A Survey," *arXiv preprint arXiv:2204.03080*, 2022.

[162] J. Han, Y. Rong, T. Xu, and W. Huang, "Geometrically Equivariant Graph Neural Networks: A Survey," *arXiv preprint arXiv:2202.07230*, 2022.

[163] A. Gupta, P. Matta, and B. Pant, "Graph Neural Network: Current State of Art, Challenges and Applications," *Materials Today: Proceedings*, vol. 46, pp. 10 927–10 932, 2021.

[164] S. Georgousis, M. P. Kenning, and X. Xie, "Graph Deep Learning: State of the Art and Challenges," *IEEE Access*, vol. 9, pp. 22 106–22 140, 2021.

[165] S. Abadal, A. Jain, R. Guirado, J. López-Alonso, and E. Alarcón, "Computing Graph Neural Networks: A Survey from Algorithms to Accelerators," *ACM Computing Surveys (CSUR)*, vol. 54, no. 9, pp. 1–38, 2021.

[166] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, and H. Liu, "Graph Learning: A Survey," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 2, pp. 109–127, 2021.

[167] Y. Xie, Z. Xu, J. Zhang, Z. Wang, and S. Ji, "Self-Supervised Learning of Graph Neural Networks: A Unified Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[168] L. Waikhom and R. Patgiri, "Graph Neural Networks: Methods, Applications, and Opportunities," *arXiv preprint arXiv:2108.10733*, 2021.

[169] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph Neural Networks: A Review of Methods and Applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[170] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph Neural Networks in Recommender Systems: A Survey," *ACM Computing Surveys (CSUR)*, 2020.

[171] C. Gao, X. Wang, X. He, and Y. Li, "Graph Neural Networks for Recommender System," in *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*, 2022, pp. 1623–1625.

[172] J. Chicaiza and P. Valdiviezo-Diaz, "A Comprehensive Survey of Knowledge Graph-Based Recommender Systems: Technologies, Development, and Contributions," *Information*, vol. 12, no. 6, p. 232, 2021.

[173] W. Jiang and J. Luo, "Graph Neural Network for Traffic Forecasting: A Survey," *Expert Systems with Applications*, p. 117921, 2022.

[174] K.-H. N. Bui, J. Cho, and H. Yi, "Spatial-Temporal Graph Neural Network for Traffic Forecasting: An Overview and Open Research Issues," *Applied Intelligence*, pp. 1–12, 2021.

[175] Z. Diao, X. Wang, D. Zhang, Y. Liu, K. Xie, and S. He, "Dynamic Spatial-Temporal Graph Convolutional Neural Networks for Traffic Forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 890–897.

[176] L. Wu, Y. Chen, K. Shen, X. Guo, H. Gao, S. Li, J. Pei, and B. Long, "Graph Neural Networks for Natural Language Processing: A Survey," *arXiv preprint arXiv:2106.06090*, 2021.

[177] M. Malekzadeh, P. Hajibabaee, M. Heidari, S. Zad, O. Uzuner, and J. H. Jones, "Review of Graph Neural Network in Text Classification," in *IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE, 2021, pp. 0084–0091.

[178] X. Zheng, Y. Liu, S. Pan, M. Zhang, D. Jin, and P. S. Yu, "Graph Neural Networks for Graphs with Heterophily: A Survey," *arXiv preprint arXiv:2202.07082*, 2022.

[179] S. Xiao, S. Wang, Y. Dai, and W. Guo, "Graph Neural Networks in Node Classification: Survey and Evaluation," *Machine Vision and Applications*, vol. 33, no. 1, pp. 1–19, 2022.

[180] J. Wang, S. Zhang, Y. Xiao, and R. Song, "A Review on Graph Neural Network Methods in Financial Applications," *arXiv preprint arXiv:2111.15367*, 2021.

[181] R. Sato, "A Survey on the Expressive Power of Graph Neural Networks," *arXiv preprint arXiv:2003.04078*, 2020.

[182] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in Graph Neural Networks: A Taxonomic Survey," *arXiv preprint arXiv:2012.15445*, 2020.

[183] P. Pradhyumna, G. Shreya *et al.*, "Graph Neural Network (GNN) in Image and Video Understanding Using Deep Learning for Computer Vision Applications," in *2nd International Conference on Electronics and Sustainable Communication Systems (ICESC)*. IEEE, 2021, pp. 1183–1189.

[184] K. Han, Y. Wang, J. Guo, Y. Tang, and E. Wu, "Vision GNN: An Image is Worth Graph of Nodes," *arXiv preprint arXiv:2206.00272*, 2022.

[185] X. Zeng, X. Tu, Y. Liu, X. Fu, and Y. Su, "Toward Better Drug Discovery with Knowledge Graph," *Current Opinion in Structural Biology*, vol. 72, pp. 114–126, 2022.

[186] T. Gaudelet, B. Day, A. R. Jamasb, J. Soman, C. Regep, G. Liu, J. B. Hayter, R. Vickers, C. Roberts, J. Tang *et al.*, "Utilizing Graph Machine Learning within Drug Discovery and Development," *Briefings in Bioinformatics*, vol. 22, no. 6, p. bbab159, 2021.

[187] P. Bongini, M. Bianchini, and F. Scarselli, "Molecular Generative Graph Neural Networks for Drug Discovery," *Neurocomputing*, vol. 450, pp. 242–252, 2021.

[188] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open Graph Benchmark: Datasets for Machine Learning on Graphs," *arXiv preprint arXiv:2005.00687*, 2020.

[189] S. Freitas, Y. Dong, J. Neil, and D. H. Chau, "A Large-Scale Database for Graph Representation Learning," *arXiv preprint arXiv:2011.07682*, 2020.

[190] Y. Zhu, Y. Xu, Q. Liu, and S. Wu, "An Empirical Study of Graph Contrastive Learning," *arXiv preprint arXiv:2109.01116*, 2021.

[191] Q. Zheng, X. Zou, Y. Dong, Y. Cen, D. Yin, J. Xu, Y. Yang, and J. Tang, "Graph Robustness Benchmark: Benchmarking the Adversarial Robustness of Graph Machine Learning," *arXiv preprint arXiv:2111.04314*, 2021.

[192] K.-H. Lai, D. Zha, J. Xu, Y. Zhao, G. Wang, and X. Hu, "Revisiting Time Series Outlier Detection: Definitions and Benchmarks," in *35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[193] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle, "On the Evaluation of Unsupervised Outlier Detection: Measures, Datasets, and an Empirical Study," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 891–927, 2016.

[194] V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking Graph Neural Networks," *arXiv preprint arXiv:2003.00982*, 2020.

[195] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of Graph Neural Network Evaluation," *arXiv preprint arXiv:1811.05868*, 2018.

[196] F. Errica, M. Podda, D. Bacciu, and A. Micheli, "A Fair Comparison of Graph Neural Networks for Graph Classification," in *International Conference on Learning Representations (ICLR)*, 2020.

[197] Q. Lv, M. Ding, Q. Liu, Y. Chen, W. Feng, S. He, C. Zhou, J. Jiang, Y. Dong, and J. Tang, "Are We Really Making Much Progress? Revisiting, Benchmarking, and Refining Heterogeneous Graph Neural Networks," pp. 1150–1160, 2021.

[198] Y. Yang, R. N. Lichtenwalter, and N. V. Chawla, "Evaluating Link Prediction Methods," *Knowledge and Information Systems*, vol. 45, no. 3, pp. 751–782, 2015.

[199] R. R. Junuthula, K. S. Xu, and V. K. Devabhaktuni, "Evaluating Link Prediction Accuracy in Dynamic Networks with Added and Removed Edges," in *IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom)*. IEEE, 2016, pp. 377–384.

[200] ——, "Leveraging Friendship Networks for Dynamic Link Prediction in Social Interaction Networks," in *12th International AAAI Conference on Web and Social Media*, 2018.

[201] S. Haghani and M. R. Keyvanpour, "A Systemic Analysis of Link Prediction in Social Network," *Artificial Intelligence Review*, vol. 52, no. 3, pp. 1961–1995, 2019.

[202] Z. Yang, M. Ding, C. Zhou, H. Yang, J. Zhou, and J. Tang, "Understanding Negative Sampling in Graph Representation Learning," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020, pp. 1666–1676.

[203] L. Backstrom and J. Leskovec, "Supervised Random Walks: Predicting and Recommending Links in Social Networks," in *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, 2011, pp. 635–644.

[204] J. Scripps, P.-N. Tan, F. Chen, and A.-H. Esfahanian, "A Matrix Alignment Approach for Link Prediction," in *19th International Conference on Pattern Recognition*. IEEE, 2008, pp. 1–4.

[205] D. Liben-Nowell and J. Kleinberg, "The Link-Prediction Problem for Social Networks," *Journal of the American Society for Information Science and Technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[206] R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla, "New Perspectives and Methods in Link Prediction," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010, pp. 243–252.

[207] L. Lü and T. Zhou, "Link Prediction in Complex Networks: A Survey," *Physica A: Statistical Mechanics and Its Applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[208] M. Maruf and A. Karpatne, "Maximizing Cohesion and Separation in Graph Representation Learning: A Distance-Aware Negative Sampling Approach," in *Proceedings of the SIAM International Conference on Data Mining (SDM)*. SIAM, 2021, pp. 271–279.

[209] B. Kotnis and V. Nastase, "Analysis of the Impact of Negative Sampling on Link Prediction in Knowledge Graphs," *arXiv:1708.06816*, 2017.

[210] J. Skardinga, B. Gabrys, and K. Musial, "Foundations and Modelling of Dynamic Networks Using Dynamic Graph Neural Networks: A Survey," *IEEE Access*, 2021.

[211] G. Xue, M. Zhong, J. Li, J. Chen, C. Zhai, and R. Kong, "Dynamic Network Embedding Survey," *Neurocomputing*, vol. 472, pp. 212–223, 2022.

[212] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic Neural Networks: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[213] C. D. Barros, M. R. Mendonça, A. B. Vieira, and A. Ziviani, "A Survey on Embedding Dynamic Graphs," *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–37, 2021.

[214] C. Wu, G. Nikolentzos, and M. Vazirgiannis, "EvoNet: A Neural Network for Predicting the Evolution of Dynamic Graphs," in *International Conference on Artificial Neural Networks*. Springer, 2020, pp. 594–606.

[215] L. Cai, Z. Chen, C. Luo, J. Gui, J. Ni, D. Li, and H. Chen, "Structural Temporal Graph Neural Networks for Anomaly Detection in Dynamic Graphs,"

in *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, 2021, pp. 3747–3756.

[216] D. Jin, S. Kim, R. A. Rossi, and D. Koutra, "On Generalizing Static Node Embedding to Dynamic Settings," in *Proceedings of the 5th ACM International Conference on Web Search and Data Mining*, 2022, pp. 410–420.

[217] C. Gao, J. Zhu, F. Zhang, Z. Wang, and X. Li, "A Novel Representation Learning for Dynamic Graphs Based on Graph Convolutional Networks," *IEEE Transactions on Cybernetics*, 2022.

[218] S. Kumar, X. Zhang, and J. Leskovec, "Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019, pp. 1269–1278.

[219] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "DyRep: Learning Representations over Dynamic Graphs," in *International Conference on Learning Representations*, 2019.

[220] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive Representation Learning on Temporal Graphs," *arXiv preprint arXiv:2002.07962*, 2020.

[221] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal Graph Networks for Deep Learning on Dynamic Graphs," *arXiv preprint arXiv:2006.10637*, 2020.

[222] Y. Wang, Y.-Y. Chang, Y. Liu, J. Leskovec, and P. Li, "Inductive Representation Learning in Temporal Networks via Causal Anonymous Walks," in *International Conference on Learning Representations (ICLR)*, 2020.

[223] Z. Cheng, X. Hou, R. Li, Y. Zhou, X. Luo, J. Li, and K. Ren, "Towards a First Step to Understand the Cryptocurrency Stealing Attack on Ethereum," in *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 2019, pp. 47–60.

[224] M. Vasek and T. Moore, "Analyzing the Bitcoin Ponzi Scheme Ecosystem," in *International Conference on Financial Cryptography and Data Security*. Springer, 2018, pp. 101–112.

[225] C. Brenig, R. Accorsi, and G. Müller, "Economic Analysis of Cryptocurrency Backed Money Laundering," in *ECIS*, 2015.

[226] T. Pham and S. Lee, "Anomaly Detection in the Bitcoin System-A Network Perspective," *arXiv preprint arXiv:1611.03942*, 2016.

[227] T. Chen, Y. Zhu, Z. Li, J. Chen, X. Li, X. Luo, X. Lin, and X. Zhange, "Understanding Ethereum via Graph Analysis," in *IEEE Conference on Computer Communications (IEEE INFOCOM)*. IEEE, 2018, pp. 1484–1492.

[228] J. Gao, W. Fan, J. Han, and P. S. Yu, "A Ggeneral Framework for Mining Concept-Drifting Data Streams with Skewed Distributions," in *Proceedings of the SIAM International Conference on Data Mining*. SIAM, 2007, pp. 3–14.

[229] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining Concept-Drifting Data Streams Using Ensemble Classifiers," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003, pp. 226–235.

[230] C. M. Bishop, *Pattern Recognition and Machine Learning*. springer, 2006.

[231] C. C. Aggarwal, *Data Classification: Algorithms and Applications*. CRC Press, 2014.

[232] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[233] G. E. Batista, A. C. Carvalho, and M. C. Monard, "Applying One-Sided Selection to Unbalanced Datasets," in *Mexican International Conference on Artificial Intelligence*. Springer, 2000, pp. 315–325.

[234] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[235] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2000, pp. 93–104.

[236] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in *8th IEEE International Conference on Data Mining*. IEEE, 2008, pp. 413–422.

[237] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler *et al.*, "API Design for Machine Learning Software: Experiences from the Scikit-Learn Project," *arXiv preprint arXiv:1309.0238*, 2013.

[238] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-Learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-365.html

[239] E. Badawi and G.-V. Jourdan, "Cryptocurrencies Emerging Threats and Defensive Mechanisms: A Systematic Literature Review," *IEEE Access*, 2020.

[240] B. E. Howell and P. H. Potgieter, "Industry Self-Regulation of Cryptocurrency Exchanges," 2019.

[241] H. Huang, W. Kong, S. Zhou, Z. Zheng, and S. Guo, "A Survey of State-of-the-Art on Blockchains: Theories, Modelings, and Tools," *arXiv preprint arXiv:2007.03520*, 2020.

[242] P. Goyal and E. Ferrara, "Graph Embedding Techniques, Applications, and Performance: A Survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[243] A. P. Motamed and B. Bahrak, "Quantitative Analysis of Cryptocurrencies Transaction Graph," *Applied Network Science*, vol. 4, no. 1, pp. 1–21, 2019.

[244] "Bitcoin Block Explorer and API," https://sochain.com/, (Accessed on 09/15/2020).

[245] "Blockchain-ETL/Ethereum-ETL," https://github.com/blockchain-etl/ethereum-etl, (Accessed on 09/15/2020).

[246] "Ethereum Scam Database," https://etherscamdb.info/scams, (Accessed on 05/14/2020).

[247] L. v. d. Maaten and G. Hinton, "Visualizing Data Using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[248] T. Pourhabibi, K.-L. Ong, B. H. Kam, and Y. L. Boo, "Fraud Detection: A Systematic Literature Review of Graph-Based Anomaly Detection Approaches," *Decision Support Systems*, vol. 133, p. 113303, 2020.

[249] W. Kudo, M. Nishiguchi, and F. Toriumi, "GCNEXT: Graph Convolutional Network with Expanded Balance Theory for Fraudulent User Detection," *Social Network Analysis and Mining*, vol. 10, no. 1, pp. 1–12, 2020.

[250] S. Kumar and N. Shah, "False Information on Web and Social Media: A Survey," *arXiv preprint arXiv:1804.08559*, 2018.

[251] P. Massa and P. Avesani, "Trust-Aware Recommender Systems," in *Proceedings of the ACM conference on Recommender Systems*, 2007, pp. 17–24.

[252] S. Sakr, A. Bonifati, H. Voigt, A. Iosup, K. Ammar, R. Angles, W. Aref, M. Arenas, M. Besta, P. A. Boncz *et al.*, "The Future is Big Graphs: A Community View on Graph Processing Systems," *Communications of the ACM*, vol. 64, no. 9, pp. 62–71, 2021.

[253] F. Monti, F. Frasca, D. Eynard, D. Mannion, and M. M. Bronstein, "Fake News Detection on Social Media Using Geometric Deep Learning," *arXiv preprint arXiv:1902.06673*, 2019.

[254] I. Chami, A. Wolf, D.-C. Juan, F. Sala, S. Ravi, and C. Ré, "Low-Dimensional Hyperbolic Knowledge Graph Embeddings," *arXiv preprint arXiv:2005.00545*, 2020.

[255] "Get Involved in Graphs 4 COVID-19," https://neo4j.com/graphs4good/covid-19/, accessed: 2021-11-04.

[256] M. Paquet-Clouston, M. Romiti, B. Haslhofer, and T. Charvat, "Spams Meet Cryptocurrencies: Sextortion in the Bitcoin Ecosystem," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, 2019, pp. 76–88.

[257] T. Zhao, X. Zhang, and S. Wang, "GraphSMOTE: Imbalanced Node Classification on Graphs with Graph Neural Networks," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021, pp. 833–841.

[258] Y. Wang, Y. Cai, Y. Liang, H. Ding, C. Wang, S. Bhatia, and B. Hooi, "Adaptive Data Augmentation on Temporal Graphs," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[259] S. Tian, T. Xiong, and L. Shi, "Streaming Dynamic Graph Neural Networks for Continuous-Time Temporal Graph Modeling," in *IEEE International Conference on Data Mining (ICDM)*. IEEE, 2021, pp. 1361–1366.

[260] W. Hu, M. Fey, H. Ren, M. Nakata, Y. Dong, and J. Leskovec, "OGB-LSC: A Large-scale Challenge for Machine Learning on Graphs," *arXiv preprint arXiv:2103.09430*, 2021.

[261] J. Shetty and J. Adibi, "The Enron Email Dataset Database Schema and Brief Statistical Report," *Information Sciences Institute Technical Report, University of Southern California*, vol. 4, no. 1, pp. 120–128, 2004.

[262] A. Madan, M. Cebrian, S. Moturu, K. Farrahi *et al.*, "Sensing the "Health State" of a Community," *IEEE Pervasive Computing*, vol. 11, no. 4, 2011.

[263] P. Panzarasa, T. Opsahl, and K. M. Carley, "Patterns and Dynamics of Users' Behavior and Interaction: Network Analysis of an Online Community," *Journal of the American Society for Information Science and Technology*, vol. 60, no. 5, pp. 911–932, 2009.

[264] M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic, and M. Wilhelm, "Bringing up OpenSky: A Large-Scale ADS-B Sensor Network for Research," in *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*. IEEE, 2014, pp. 83–94.

[265] S. Huang, Y. Hitti, G. Rabusseau, and R. Rabbany, "Laplacian Change Point Detection for Dynamic Graphs," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020, pp. 349–358.

[266] J. H. Fowler, "Legislative Sponsorship Networks in the US House and Senate," *Social Networks*, vol. 28, no. 4, pp. 454–465, 2006.

[267] G. K. MacDonald, K. A. Brauman, S. Sun, K. M. Carlson, E. S. Cassidy, J. S. Gerber, and P. C. West, "Rethinking Agricultural Trade Relationships in an Era of Globalization," *BioScience*, vol. 65, no. 3, pp. 275–289, 2015.

[268] E. Voeten, A. Strezhnev, and M. Bailey, "United Nations General Assembly Voting Data," 2009. [Online]. Available: https://doi.org/10.7910/DVN/LEJUQZ