Exploring MempoolScape: Bitcoin Mempool Logs Dataset Dec‘20-Feb‘21

Haseeb Saeed[1], Asad Salman[2], Muhammad Anas Tahir[3], and Syed Taha Ali[4]

[*]National University of Sciences and Technology Islamabad, Pakistan
[1]hsaeed.bscs17seecs@seecs.edu.pk
[2]me@asad.co
[3]mtahir.bscs17seecs@seecs.edu.pk
[4]taha.ali@seecs.edu.pk

Abstract - Proof-of-work cryptocurrencies like Bitcoin function within an intricate network where transactions compete for block inclusion. We present MempoolScape, a unique dataset encapsulating about 26.1 million Bitcoin transactions with 18 distinct features. Unlike any existing public dataset, MempoolScape offers an in-depth snapshot of the Bitcoin mempool at any specific block height, enabling nuanced analyses beyond mere transaction logs. This paper focuses on the methodologies employed for data collection, the variety and scope of features, and the potential applicability of MempoolScape. It serves as a versatile instrument for analysts, scholars, and decision-makers focused on understanding the complexities of the Bitcoin network.

I. Introduction

Bitcoin has fundamentally altered the landscape of digital finance by introducing a decentralized currency system based on blockchain technology [1]. The network comprises various components, including miners, users, and a crucial but often overlooked element known as the mempool [2], [3]. The mempool's dynamics significantly influence critical aspects of the Bitcoin network, such as transaction efficiency, associated costs, and overall user experience [4].

However, the importance of the mempool extends beyond mere logistical functions. A deeper understanding of this transient storage area for unconfirmed transactions has far-reaching implications for academic inquiry and practical applications. It can be instrumental in predicting transaction latency, estimating transaction fees, and optimizing network operations. As such, the focus of this paper is to advance the understanding of Bitcoin's operational intricacies and contribute to more effective network utilization.

To provide a comprehensive view, this paper is organized into various sections that will describe the nature of the dataset under examination, elaborate on the methodologies employed for data collection, detail the specific features contained within the dataset, and finally, discuss the potential applications and value that this dataset brings to the broader context of Bitcoin network analysis.

II. Data

Features in this data set are extracted by modifying the Bitcoin core, specifically `src/txmempool.cpp`, using code written in C++ . This modification enabled us to log all mempool activity, including all transaction entries and exits, in JSON format. These transactions were processed (more details in the Data Collection Process section). Further features were derived using the extracted features, and a few others were imported. The data set is structured in CSV format where each row is an transaction and each column represents an attribute of that transaction. While primarily aimed at creating snapshots of the network state at any height, the dataset also enables latency prediction for unconfirmed transactions and supports various other applications. New features can be easily added, enhancing its utility.

A. Data Specification

Subject area	Computer Science
Specific subject area	Cryptocurrencies, Artificial Intelligence, Machine Learning
Type of Data	Table in CSV format
How data was acquired	Data was gathered by modifying the Bitcoin to log transactions, and additional features incorporated from external APIs
Data format	Processed
Data source location	Pakistan, Cognet Lab, SEECS, NUST, Islamabad
Data accessibility	The data set is available for download on Kaggle (kaggle.com/datasets/itshaseebsaeed/bitcoin-mempool-state-data) under the CC BY-SA 4.0 license
Temporal Coverage Dates	12/04/2020 to 02/26/2021

B. Data Collection Process

The data acquisition process began with enabling the logging [5] of all incoming transactions in an ‘EntryLogTimestamp‘ file, and outgoing or accepted transactions in an ‘ExitLogTimestamp‘ file. This approach facilitated the matching of incoming transactions with accepted ones, using their hashes.

Each matched transaction was recorded with both entry and exit times. Additional external features were also incorporated, such as the BTC price and Mining Pool's Profitability [6], these features were fetched from external API [7] and then mapped onto the transactions using timestamps. A few features like Offset, Transaction Fee Per Kilobyte, Blocks To Confirm, Seconds Since Last Block, and Count on Entry Time were derived from the base features [8]. Ultimately, any redundant features were discarded, resulting in the final data set.

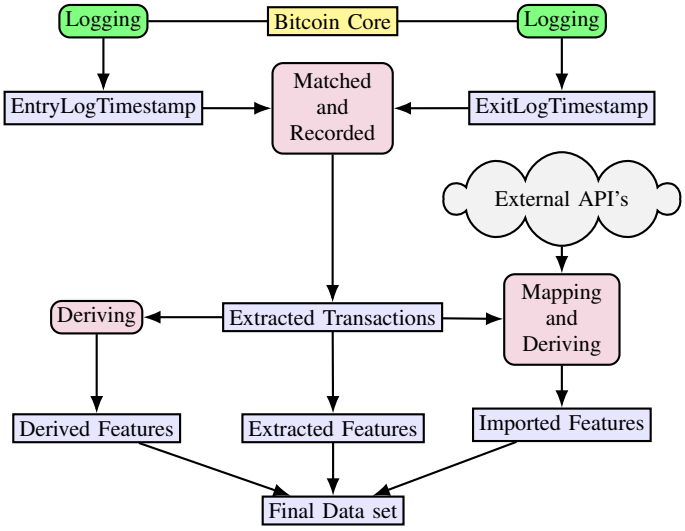

Fig. 1: Data extraction scheme

C. Feature Selection

This section outlines the features selected to train the machine learning model. A total of 18 features were narrowed down, which belong to different components of the network: features related to transaction, block, mempool, mining pool, economy, and network.

Our feature set $\mathbb{F}$ is formed by both Extracted ($\mathbb{E}$), Derived ($\mathbb{D}$), and Imported ($\mathbb{I}$) features,

$$\mathbb{F} = \mathbb{E} \cup \mathbb{D} \cup \mathbb{I}$$

1) Extracted Features ($\mathbb{E}$): These features were extracted directly from the raw transaction data.

t_s	TxSize	Transaction size in bytes. Greater the size of the transaction, higher the fee required for acceptance by miners. Miners have limited block space of 1 MB and prefer to fit transactions to maximize profit.
t_f	TxFee	The total fee associated with this transaction, denoted by t_f. As rational entities, miners are likely to select transactions with higher fees as the system scales.
t_{in}	TxIn	Number of incoming Unspent Transaction Outputs (UTXOs) contributes to the overall size of the transaction. A larger transaction size necessitates additional verification work by miners.
t_{out}	TxOut	Number of out-going UTXOs, also increasing the size of the transaction and more verification work for miners.
d_c	DescendantCount	A descendant transaction is one that is part of a chain of transactions where each transaction spends the output of the previous one. The term "descendant" is used to describe transactions that are "children", "grandchildren"', "great-grandchildren", etc., of a specific transaction. This feature represents the Total number of descendants.
d_f	DescendantFee	Total fee paid by the descendants.
d_s	DescendantSize	Total size of descendant in bytes
en_h	EntryBlock	Block height transaction entered the mempool.
ex_h	ExitBlock	Block height transaction left the mempool.
en_t	EntryTime	Unix Time when transaction entered the mempool.
ex_t	ExitTime	Unix Time when transaction exited the mempool.

2) Derived Features ($\mathbb{D}$): These features were derived from the base attributes, utilizing both incoming and outgoing time data to construct a snapshot of the mempool.

ρ	TxFeePkb	The total fee paid per kilobyte is a significant metric in the transaction process. The higher the ratio, the greater the chances of a transaction being accepted.
Δb_h	BlocksToConfirm	Number of blocks required for transaction confirmation.
Δt_{ep}	SecSinceLB	This metric signifies the time, in seconds, that has elapsed between the transaction timestamp and the most recent block timestamp. As time increases, it creates mounting pressure on miners to produce the succeeding block. Concurrently, users may become increasingly inclined to offer higher transaction fees to secure a spot in the next available block.
δ	Offset	Represents the offset in bytes. It quantifies, for each transaction, the number of bytes already occupied by transactions with higher fees-per-kilobyte that have not yet been approved, within the context of a hypothetical next block.
C_n	CountOnEntryTime	Number of transactions in the mempool when transaction entered.

The elements in $\mathbb{D}$ are derived. For example fee density ρ for a transaction t is determined using the following equation:

$$\rho_t = \frac{t_f}{t_s} \tag{1}$$

Blocks to confirm Δb_h is derived by subtracting the ExitBlock ex_h with EntryBlock en_h.

$$\Delta b_h = ex_h - en_h \tag{2}$$

Some features are more complex and challenging to derive, such as Δt_{e_p}, δ, and $\mathbf{n}$. When training the model, it's necessary to introduce new concepts to provide context for each transaction at the time it was initiated. One such feature is Δt_{e_p}, which is computed as shown below. Here, $Bs_t^{(i)}$ symbolizes

either the successor or predecessor of a transaction t. If $i = 0$, the successor is the first block epoch B_{e_p} mined after the inception time t_{e_p} of t. Conversely, if $i = -1$, the predecessor is the first preceding B_{e_p} of t_{e_p}:

$$\begin{aligned} &\Delta t_{e_p} = t_{e_p} - Bs_t^{(-1)} \\ \text{where} \quad &Bs_t^{(-1)} \leq t_{e_p} \leq Bs_t^{(0)} \end{aligned} \tag{3}$$

To explain the concept of the **offset** δ, we first define a set of transactions denoted by S. Within this set, each offset δ_S is an incrementing number that starts from 0 and ranges up to the maximum value of B_{h_e}. The set S encompasses all transactions for which the entry time t_{e_p} falls within a specific time span, ΔBs_t. These transactions in set S are arranged according to their fee density, ρ, and this ordering helps in defining the offset, δ.

$$\begin{aligned} &S = (t^{(0)}, t^{(1)}, \ldots, t^{(n)}) \\ &\text{where } \rho^{(0)} < \rho^{(1)} < \cdots < \rho^{(n)} \\ &\text{and } Bs_{tj}^{(-1)} \leq t_{ep}^{(j)} < \cdots < Bs_{tj}^{(0)} \text{ for } j = 1, 2, \ldots, n \end{aligned} \tag{4}$$

In the above equation, for the sake of clarity, $t^{(j)}$ is equivalent to t_j, and $B_{t_j^{(i)}}$ symbolizes either the successor or predecessor for the transaction $t^{(j)}$. The following equation illustrates the offset for a transaction $t^{(n)}$ that is part of the set S:

$$\delta_s^{(n)} = \begin{cases} t_q^{(j)} & \text{if } j = 0, \\ \delta_s^{(j-1)} + t_q^{(j)} & \text{for } j = 1 \text{ to } n. \end{cases} \tag{5}$$

This mathematical representation provides a comprehensive understanding of how the offset δ is calculated and its significance in the context of the transactions within the set S.

To explain the concept of the **Count on Entry Time $\mathbf{C_n}$**, we must consider the state of the mempool at the exact moment a specific transaction enters. The $\mathbf{n}$ represents the total number of transactions present in the mempool at that time, providing a snapshot of the network's activity or congestion level. We're using S and ΔBs_t just like offset. Let's define:

$$\begin{aligned} &S = \left(t^{(0)}, t^{(1)}, \ldots, t^{(n)}\right) \\ &\text{where } Bs_{tj}^{(-1)} \leq t_{ep}^{(j)} < \cdots < Bs_{tj}^{(0)} \text{ for } j = 1, 2, \ldots, n \end{aligned} \tag{6}$$

C_n = Number of transactions in the mempool when transaction entered

This value can be expressed as:

$$\begin{aligned} C_n = \sum_{i=1}^{n} t_i \quad \forall \quad S_k \\ \text{where } 0 \leq k \leq \max(B_{h_e}) \end{aligned} \tag{7}$$

where t_i represents each individual transaction in the mempool at the entry time of the specific transaction under consideration, and n is the total number of transactions in the mempool at that moment.

3) Imported Features ($\mathbb{I}$): The features under consideration were obtained from external Application Programming Interfaces (APIs) specialized in blockchain data. These values were subsequently mapped onto the corresponding transactions using their respective timestamps for alignment.

PPF	MiningPoolProfitability	Measure of USD/Day for 1 THash/s, affecting miners' interest in higher fees. This feature is calculated when transaction entered the mempool.
BTC_p	BtcPrice	Value of Bitcoins in US Dollars. This feature is refer to Bitcoin price at the time of entry.

D. Value of the data

The information contained within the data set offers significant insights into the behavior and characteristics of Bitcoin transactions. By analyzing this data, several key applications and benefits can be realized:

- **Optimizing Transaction Fees:** Informs fee selection for cost-effective, timely transaction processing [9].
- **Predicting Transaction Latency:** Enables Transaction latency forecasting models, crucial for both individual and business transactions [10].
- **Enhancing Network Efficiency:** Assists miners and network participants in enhancing Bitcoin network efficiency [11].
- **Node Comparison:** Enables comparisons between different nodes, such as contrasting the mempool states across them [12].
- **Time-Specific Visualization:** Allows to reconstruct time-specific snapshots for visualizing the network perspective of a particular node
- **Anomaly Identification:** Aids in anomaly detection for debugging and highlighting suspicious activities [13].
- **Forensic Investigations:** Useful for forensic probes into incidents like the 2015 "dust" attack [14].
- **Community Policy Impact:** Supports studies on the effects of community policies like pay-per-fee [15].
- **Client-Side Enhancements:** Contributes to developing and testing client-side security measures [16].

E. Predicting Transaction Latency

Forecasting in this context refers to predicting the amount of time or the number of blocks required for a transaction to be confirmed. This offers valuable insights into network efficiency and performance, aiding users in optimizing both their fee expenses and approval times.

Each derived feature serves a unique purpose in creating robust predictive models for transaction latency. The feature TxFeePkb(ρ) measures the urgency of a transaction, serving as a strong indicator for quick confirmations. BlocksToConfirm(Δbh) can act either as a target variable for latency predictions or be used alongside other features in multi-output regression models. The feature SecSinceLB(Δt_{ep}) captures the 'pressure' on the system by measuring the time elapsed since the last block, making it

a useful input for understanding system state at the time a new transaction is made. Similarly, Offset(δ) accounts for the already occupied space in the mempool, helping the model gauge how 'crowded' the next block might be. Lastly, CountOnEntryTime(C_n) provides context about the competition for block space at the time the transaction was made, making it another essential input feature.

Together, these features can enhance the model's ability to predict transaction latency, offering a more reliable and dynamic forecasting method compared to traditional fee estimators which are often susceptible to manipulation.

1) Feature Selection for Forecasting: The dataset comprises an extensive feature set $\mathbb{F} = \mathbb{E} \cup \mathbb{D} \cup \mathbb{I}$, not all of these features are applicable for the specific task of latency forecasting. In the context of building a predictive model for transaction latency, it is crucial to identify and select features that provide relevant insights without introducing any data leakage. Specifically, features such as EntryBlock (enh), ExitBlock (exh), EntryTime (ent), and ExitTime (ext) should be excluded from the model. The target variable for training will be BlocksToConfirm (Δbh), which represents the number of blocks required for transaction confirmation. This choice aligns with the primary focus of forecasting latency within the Bitcoin network. Furthermore, while the primary goal of this data set is latency forecasting, it is worth noting that the versatile nature of the data allows for other applications and the derivation of numerous new features[17] like Size on Entry Time and Transaction count on last block e.t.c.

F. Data Analysis

The data set provided a detailed view into the workings of the Bitcoin network. Key metrics include an average transaction size of 595.64 Bytes and a mean fee of $36,067.52$ satoshis. Most strikingly, the dataset records a maximum transaction fee of $349,079,600$ satoshis, illuminating the extreme volatility and the intricate fee dynamics within the Bitcoin ecosystem. The data set also tracks the average number of blocks required for confirmation, 13.15, and the variability in Bitcoin's price, which ranges from $\$17,632.00$ to $\$58,317.00$. These comprehensive illuminates the fluidity of the Bitcoin market, potentially signaling broader economic trends.

1) Features Analysis: This section analyze a few important features in the data set.

- **Blocks To Confirm Distribution :** The distribution of `BlocksToConfirm` highlights key patterns within the Bitcoin network's transaction confirmation process. From the logarithmic scale plot, it's apparent that the majority of transactions require a relatively small number of blocks for confirmation, with the peak density occurring around a value of $\log(\text{BlocksToConfirm} + 1) \approx 1$. This suggests that most transactions are confirmed within just a few blocks. However, the long tail extending towards higher values indicates that there are cases where confirmation can take significantly longer, up to approximately $\log(\text{BlocksToConfirm}+1) \approx 4$. This pattern might reflect scenarios of network congestion or lower transaction fees leading to delayed processing.

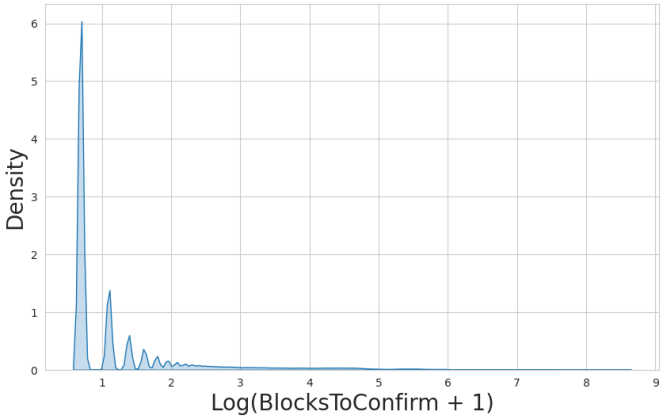

Fig. 2: Logarithmic Distribution of Transaction Confirmation Latency in Blocks

- **TimeToConfirm Distribution :** The distribution of the `TimeToConfirm` variable, representing the time required for Bitcoin transactions to be confirmed, exhibits a skewed right shape with some multimodality. The curve reveals notable characteristics, with a pronounced peak at approximately $\log(\text{TimeToConfirm} + 1) \approx 6.52$, corresponding to a confirmation time of around 11.29 minutes. This area likely represents the typical behavior within the Bitcoin network, where transactions are swiftly processed. Conversely, the long right tail extending to higher values and the presence of smaller secondary peaks indicate a more complex underlying structure. These instances could correspond to periods of high network activity or transactions with lower fees, leading to prolonged confirmation times.

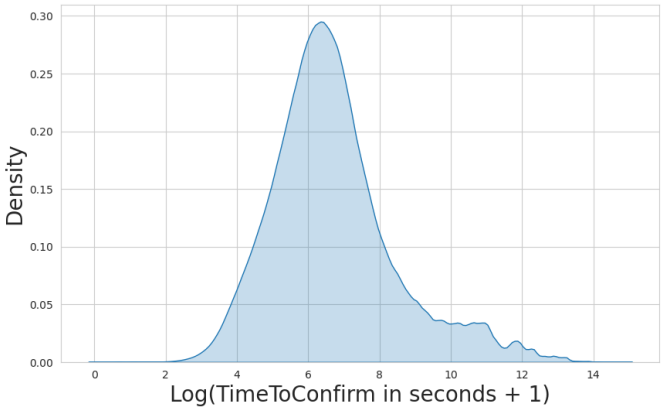

Fig. 3: Logarithmic Distribution of Transaction Confirmation Latency in Seconds

2) Timespan Analysis: Timespan analysis involves the study of trends and patterns within a specific period to gain insights into underlying dynamics and behaviors. Throughout

the analyzed timespan, the Bitcoin price exhibited a general upward trend. Concurrently, the transaction count showed a tendency to increase, especially within certain price ranges, and remained within a specific range for most of the period.

- **Bitcoin Price vs Transaction Volume:** The plot depicting the relationship between the Bitcoin price (BTC_p) and the transaction count on entry time (C_n) reveals a positive relationship, although not strongly linear, between the two variable. The few outlier points emphasize the complexity of this relationship, warranting further investigation to understand the underlying mechanisms.

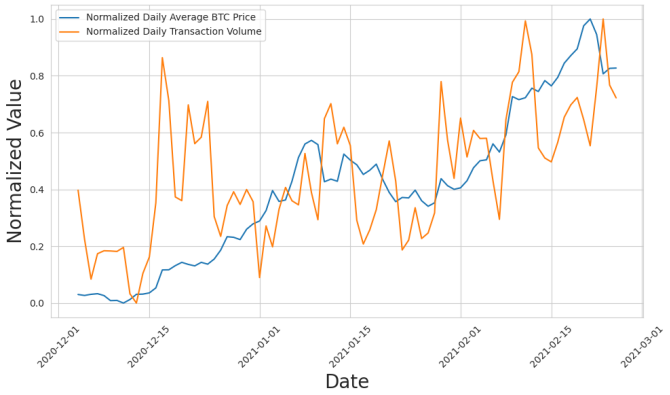

Fig. 4: Comparison of Normalized Daily Average Bitcoin Price and Transaction Volume at Entry Time

- **Transaction Fee vs Volume:** The plot captures the time-based fluctuations of normalized daily transaction volume and average fees. Both metrics generally trend upwards, punctuated by periodic peaks and troughs.There are evident periodic fluctuations in both curves, with noticeable peaks and troughs. Though they largely move in sync, indicating a positive correlation, occasional divergences are observed. Such cycles may relate to specific events, market shifts, or recurring user behaviors like weekly or monthly transaction patterns.

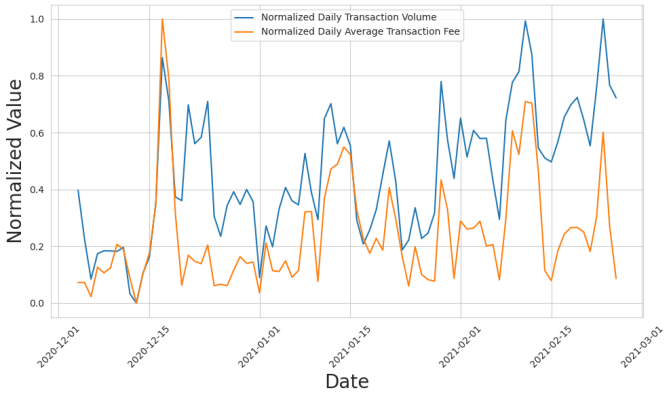

Fig. 5: Comparison of Normalized Number of Transactions in Mempool at Entry Time and Transaction Fee

III. Conclusion

The value of the data set extends beyond mere observation and analysis. It opens up avenues for practical applications that can enhance the functionality, efficiency, and user experience within the Bitcoin network. Whether for individual users seeking to optimize their transactions or for industry stakeholders aiming to innovate and improve the system, the information serves as a foundational resource that drives informed decision-making and technological advancement.

References

[1] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[2] C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, 2013.

[3] Peter R. Rizun. A transaction fee market exists without a block size limit. 2016.

[4] Möser M. Shahandasti S.F. Hao F. McCorry, P. Towards bitcoin payment networks. *In: Liu, J., Steinfeld, R. (eds) Information Security and Privacy. ACISP 2016. Lecture Notes in Computer Science(), vol 9722. Springer, Cham. https://doi.org/10.1007/978-3-319-40253-6*$_4$, 2016.

[5] Navneet Kumar Pandey, Kaiwen Zhang, Stéphane Weiss, Hans-Arno Jacobsen, and Roman Vitenberg. Distributed event aggregation for content-based publish/subscribe systems. In *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, pages 95–106, 2014.

[6] Bikramaditya Datta and Idan Hodor. Cryptocurrency, mining pools' concentration, and asset prices. 2023. Available at SSRN: https://ssrn.com/abstract=3887256 or http://dx.doi.org/10.2139/ssrn.3887256.

[7] Katembo Ezéchiel, Shri Kant, and Dr Agarwal. A systematic review on distributed databases systems and their techniques. *Journal of Theoretical and Applied Information Technology*, 96, 01 2019.

[8] Vijay Kumar and Sarat Kumar Patra. *Feature Engineering for Machine Learning and Deep Learning Assisted Wireless Communication*, pages 77–95. Springer International Publishing, Cham, 2021.

[9] Enrico Tedeschi, Tor-Arne S. Nordmo, Dag Johansen, and Håvard D. Johansen. On optimizing transaction fees in bitcoin using ai: Investigation on miners inclusion pattern. *ACM Trans. Internet Technol.*, 22(3), jul 2022.

[10] Enrico Tedeschi, Tor-Arne S. Nordmo, Dag Johansen, and Håvard D. Johansen. Predicting transaction latency with deep learning in proof-of-work blockchains. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4223–4231, 2019.

[11] Mengya Li, Yang Qin, Bing Liu, and Xiaowen Chu. Enhancing the efficiency and scalability of blockchain through probabilistic verification and clustering. *Information Processing Management*, 58(5):102650, 2021.

[12] Sehyun Park, Seongwon Im, Youhwan Seol, and Jeongyeup Paek. Nodes in the bitcoin network: Comparative measurement study and survey. *IEEE Access*, 7:57009–57022, 2019.

[13] Youssef Elmougy and Oliver Manzi. Anomaly detection on bitcoin, ethereum networks using gpu-accelerated machine learning methods. In *2021 31st International Conference on Computer Theory and Applications (ICCTA)*, pages 166–171, 2021.

[14] Yan Wu, Fang Tao, Lu Liu, Jiayan Gu, John Panneerselvam, Rongbo Zhu, and Mohammad Nasir Shahzad. A bitcoin transaction network analytic method for future blockchain forensic investigation. *IEEE Transactions on Network Science and Engineering*, 8(2):1230–1241, 2021.

[15] Peter Howson and Alex de Vries. Preying on the poor? opportunities and challenges for tackling the social and environmental threats of cryptocurrencies for vulnerable and low-income communities. *Energy Research Social Science*, 84:102394, 2022.

[16] Katharina Krombholz, Aljosha Judmayer, Matthias Gusenbauer, and Edgar Weippl. The other side of the coin: User experiences with bitcoin security and privacy. pages 555–580, 05 2017.

[17] Bhavani Thuraisingham. Blockchain technologies and their applications in data science and cyber security. In *2020 3rd International Conference on Smart BlockChain (SmartBlock)*, pages 1–4, 2020.