IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 8, August 2022

DOI: 10.17148/IJARCCE.2022.11823

A Novel Temporal CNN Model to Predict Malicious Transactions in Ethereum Blockchain

Mohammed Baz

Department of Computer Engineering, College of Computer and Information Technology, Taif Univer-sity, Taif, 21994, Saudi Arabia, mo.baz@tu.edu.sa

Abstract: Blockchain is one of the most advanced technologies that play an important role in many different fields such as healthcare, capital markets and logistics. Among the many existing blockchain platforms, the integration of the Turing-complete virtual programming engine with the blockchain makes the Ethereum blockchain one of the most paramount infrastructures for various types of applications, including but not limited to cryptocurrency trading, smart contracts, decentralised finance and metaverse. Nevertheless, Ethereum like many other computing systems, has fallen victim to vector attacks that exploit its vulnerabilities and have catastrophic consequences. Out of the need to protect Ethereum from such attacks, this paper proposes a novel deep learning model based on convolutional neural networks. The proposed model treats the transaction, which is the atomic entity in this platform, as a stochastic time series and then develops two specific task layers that are compatible with the traditional CNN architecture. The first layer is responsible for detecting the seasonal characteristics of the transactions, while the second layer is used for detecting the trend. These two layers are integrated with the traditional architecture to form a powerful temporal CNN architecture that can classify different types of attacks. The performance of the proposed model was evaluated from a different perspective using real transactions collected from the Ethereum main-net network. The results of the comprehensive evaluations show the ability of the proposed model to perfectly identify malicious transactions in the Ethereum blockchain.

Keywords: Ethereum blockchain, sandwich attack, front-running, block stuffing, temporal CNN.

I. INTRODUCTION

A blockchain can be abstracted as a chronologically linked list of cryptographically secured elements called blocks that are replicated on a collection of devices connected via a peer-to-peer network [1]. The content of the blocks varies depending on the purpose the blockchain serves: in cryptocurrency-based blockchains such as Bitcoin [2] and Ethereum [3], the block contains transactions that take place between parties, while in smart contract-based blockchains such as Ethereum and Tezos[4], the blocks stores pieces of code, while in non-fungible token (NFT) based blockchains such as Ethereum, the block content is metadata of the intellectual property of the assets. This great diversity, combined with the lack of need for a trusted third party, immutability to change, transparency and a high degree of fault tolerance, facilitates the use of blockchain in various sectors, including healthcare, capital markets and logistics[1,5].

Ethereum is one of the most prominent blockchain platforms, that was devised to offer more than just a trading platform for cryptocurrencies. Specifically, Ethereum equips the traditional blockchain architecture with a Turing-complete programming virtual machine dubbed Ethereum Virtual Machine (EVM) that can store and execute pieces of code. This in turn ushers in a new era of applications, including smart contracts, Decentralized autonomous applications (DAOs), Decentralized Finance (DeFi) and Metaverse[1,2,5]. Ethereum analytics show its prominence; according to[5-6], there are around 120 million Ethereum tokens in circulation, with a market capitalisation of $123.90 billion. In the first quarter of this year, 105.58 million transactions were made, with an average growth rate of 308.0%, while the total number of smart contracts created in the same period was 1.45 million, an increase of 24.7% over the number of smart contracts created in the fourth quarter of 2021. As for the number of users, several works report that the average daily number of active addresses on this blockchain is 329,900, while the number of active traders of Ethereum NFTs reached 354,000. The market analysis of DAO shows the dominance of Ethereum compared to its other platforms; about $8.2 billion out of $9.5 billion held in DAO Treasuries is based on Ethereum. Inspired by these readings this paper uses Ethereum as an example for the proposed algorithm.

One of the important security countermeasures defined by the legacy version of Ethereum is to associate each transaction (a transaction is an atomic entity of a block that requests a change to the Ethereum state machine) with a fee called a gas. In principle, this fee is defined based on the computational complexity of the codes contained in the transaction and is paid by the account from which the transaction originates to the miners (accounts with hash power that can append the new blocks to the Ethereum chain). From a theoretical perspective, the proportional relationships between the computational complexity of transactions and monetary costs make sending computationally intensive operations (such as hostile infinite loops) extraordinarily expensive, which in turn reduces the possibility of triggering DoS attacks.

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 8, August 2022
DOI: 10.17148/IJARCCE.2022.11823

Moreover, paying fees encourages miners to include new blocks as quickly as possible, not only accelerates blockchain convergence but also mitigates sybil attacks.

However, it has been evident from several incidents that the simple transaction fee scheme defined in the legacy version is incapable to achieve the desired goals. Omnipresent sandwich attack [7] is one such attack that exploits the relationships between the gasPrice and the transaction order to sandwich the victim transaction be-tween front-running and back-running fake transactions that aim to seize the trading opportunities. According to [8], at least 64217 ETHs worth USD 189,311,716 were syphoned from Uniswap, one of the largest decentralized exchange platforms (DEX), between May 2020 and May 2021. The impact of the attack becomes even more serious when dishonest miners are involved in such attacks. Basically, dishonest miners attempt to make illegal profits by scanning the outstanding transactions and then changing the order and/or inclusion of other transactions by inserting self-profitable transactions with higher gasPrice. The term Maximum Extractable Value (MEV) was coined to measure the amount of profit made in this way. The value of MEV since 1 January 2020 has been estimated at $664,003,762[9], while [10] shows that more than 90% of the total amount of MEV is spent in gas fees. Worth still, some work shows that miners can tamper the blockchain by conducting a hard fork to make more profit, as in the case of the fee-based forking attack[11] , or rewrite the blockchain history to steal funds from previous smart contracts, as in the Time Bandit attack[] . Another class of attack vectors, in which miners collude together or are even bribed by other users to work to their own advantage, has been described in some papers [12].

To defend against this type of attack, there is a need to develop a robust model that can identify the malicious transactions as soon as adversaries submit them and before they are validated and mined. However, developing such a model is a multi-faceted challenge. One of these challenges is the large volume of transactions submitted to the Ethereum main-net; according to [5], about 1.048 million transactions were submitted every day this year. The second challenging aspect is related to the stochastic nature of transactions and their high temporal correlation. In fact, most of the most severe attacks, such as sandwich attacks and time bandit attacks, are mainly based on manipulating the time at which transactions are executed. The last aspect to consider when developing a predictive model for malicious transactions is the rapid evolution of the techniques used to carry out such attacks; some examples are sniper bots that scan the transactions, DUSTBot that floods the blockchain with fraudulent transactions, or crypto-jacking bots that compromise victims' devices and use them to mine transactions in favour of the adversary.

Motivated by the importance of identifying malicious transactions in ensuring the stability and security of blockchain networks; this work uses an artificial neural network (ANN) to develop deep learning model that can predict malicious transactions. Amongst the widely used ANN architectures that appear in the open literature, such as MultiLayer Perception (MLPs), Recurrent Neural Networks (RNNs), Graph Neural Networks and Convolutional Neural Networks (CNNs)[13], the CNNs architecture is used in this work to develop the proposed model. This choice is justified in two ways: firstly, by the ability of CNNs to automatically extract the patterns required for predictions from the raw input data (i.e., the transaction stream) without the need to a priori match the dimension of the CNN input layer to the dimension of the data, as is the case with other ANN architectures such as MLP. Specifically, the foremost layer in a CNN, i.e. the convolutional layer, applies a filters bank of predefined dimensions to the inputs to determine their common distinguishing features. Moreover, in CNNs, multiple convolutional layers can be superimposed to intensify feature extraction at deeper levels. This is of tremendous importance for the development of an effective malicious transaction detector, as it facilitates the detection of temporal features of transactions (e.g. seasonality, trends and/or stationarity) under different temporal variations without the need to recreate a special feature extractor for each type of transaction or to use a preprocessing mechanism. Albeit RNNs can also function as feature extractors and are widely used in the field of temporal prediction, the sparse number of connections of the CNN is the second reason for choosing the CNN in this study. The sparse connections of the CNN result from the ability of each layer to concisely pass on its results to the next layer using smaller parameter spaces. That is, the convolutional layer summarises the most important features found in the input into a smaller dimension known as the feature map; similarly, the next layer, known as the pooling layer, further summarises the feature map before passing it on to the subsequent output layer. On the other hand, the core concept on which RNNs are built is to recurrent the results of one layer into other layers so that the temporal relationships of the input can be revealed sequentially. This in turn increases the parameter spaces relative to the depth of the network and introduces the possibility of the vanishing gradient problem, making the model impractical for large data sets, as is the case with Ethereum transactions. However, considering the fact that the traditional CNN architecture was designed to process image data and not time-series data, it is clear that this architecture needs to be fundamentally changed in order to take full advantage of this architecture for the development of the malicious-transaction identification model. The main contribution of this work can be summarised as follows:

1- Developing a novel CNN-based architecture that suits the temporal characteristics of Ethereum transactions. The main assumption in developing these layers is that transactions are stochastic and therefore the best technique for predicting their future behaviour is to decompose them into elementary components. Inspired by the principles of time series decomposition, which shows that a time series can be decomposed into two main components: seasonal

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 8, August 2022
DOI: 10.17148/IJARCCE.2022.11823

and trends, two different layers are proposed: (i) the seasonal-detect layer, whose main responsibility is to detect the fractal parts of the time at which malicious transactions occurred. (ii) the trend-discovering layer, which is used to reveal the bias that dominates the underlying process by which the transactions originated. All these layers are built according to the principles of the CNN architecture, which in turn allows them to interact and interleave to form deep networks.

2- Conducting comprehensive evaluations of the proposed model from different angles using real transactions collected on the Ethereum main-net and then validating them against various datasets published in peer works. These evaluations are measured using standard statistical metrics such as True Positive, False Positive, True Negative False Negative, Accuracy, Recall and F1-Score. The results of these evaluations show that our proposal is able to predict malicious transactions with a high level of accuracy.

The reminder of this paper is organised as follows: Section 2 reviews related works that have appeared in the open literature, and Section 3 describes the underlying transaction fees that is used in the Ethereum block chain as well as the structure of the datasets used in this study. The detailed structures of the proposed model are described in Section 3, while the results and discussion are presented in Section 4 and Section 5 concludes the work.

II. RELATED WORKS

The work presented in [14] uses a rule-based heuristic approach to identify the following types of frontrunning attacks: Displacement, Insertions and Suppression. In this work, the displacement attack is assumed to occur when two or more transactions with similar contents but different gas price coincide. The insertion at-tack takes place when a token-transferring transaction is sandwiched between two trans-actions whose aim is to trade approximately the same amount of tokens but whose gas prices are higher and lower than the gas price of that transaction respectively. Finally, it is assumed that the suppression forntrunning attack happens when the same account initiates a large number of highly computationally intensive transactions that are spread across multiple blocks. This work, uses a combination of Bloom and linear filters as a means to apply these rules to a dataset containing 11 thousand blocks with about 1 billion transactions from a five-year period starting in 2015. The results reported by the authors show that their model is able to identify about 200,00 incidents with more than USD 18.41 million in profit for the attackers, of which USD 300,000 goes to the miners in the form of transaction fees. In addition, this work points out that due to the need to maintain the traceability of the proposed model, some rules describing the attacks are relaxed, which in turn leads to some cases being overlooked.

Motived by the benefits that can be reaped from employing a data-driven approach to detect frontrunning attacks; the work appeared in [15] proposes a binary classifier based on artificial neural networks. This model uses the Long Short-Term Memory (LSTM) architecture to predict the gas price of the nor-mal transaction based on the gas price of the previous 12 normal transactions. The output of this architecture, along with another 6 statistical features describing the historical characteristics of the gas prices, are fed into the multilayer perceptron (MLP) architecture, from which the class of the transaction is obtained. In order to demonstrate the suitability of the proposed model for the real environment, the weighting matrices of this model are extracted and then used to develop the so-called attack detector smart contract. The accuracy of the proposed model was evaluated by measuring its ability to classify different types of attacks presented in the dataset developed by [14]. The results of these evaluations show that the displacement, insertion and suppression front-running attacks can be identified with an overall accuracy of 88.85, 89.08 and 85.53, respectively. It also shows that the time taken to invoke the attack detector smart contract is about 3 seconds. However, the main drawback of this model is that it is a supervised learning approach, which means that it can only detect the types of attacks whose behaviour has been previously identified.

The work presented in [16] proposes an unsupervised deep learning model to detect anomalous transactions on the Ethereum blockchain. The proposed model uses a sequence-to-sequence re-current autoencoder (RAE) architecture, in which the encoder network is trained to capture the intrinsic patterns of normal traffic and then compress them into a lower dimension space. The output of the encoder is then fed into the decoder network, which attempts to generate the full resolution samples from the compressed patterns. The anomaly score is then assigned by comparing the actual input with its regenerated version. The proposed model uses 32 Long Short Term Memory (LSTM) cells to perform the encoding and de-coding processes, and the median of the regenerated errors as the threshold for the anomaly. In this work, the transactions and gas values of the blocks are used as the main input for the encoder network. These features are extracted from the Ethereum Classic Block-chain (ETC), which spans a period of four years from 2015. The authors of this work later improved this approach by adopting the sequence-to-sequence ensemble with snapshot training scheme in the lieu of the conventional autoencoder architecture[17]. This improvement was justified by the need for a more flexible architecture that can codify the concept drift and progressively in its learnable parameters. The results reported in the two papers show that they are able to predict the DAO[18] and 51% attacks that occurred in 2016 and 2019,

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 8, August 2022
DOI: 10.17148/IJARCCE.2022.11823

respectively. However, no quantitative measure of performance in terms of standard statistical metrics is provided in these papers.

Using Graph-based Deep Learning (GDL) models to predict anomalies in the Ethereum network is one of the research directions that has been appeared in the open literature. Most of these works adopt the account-centric representation, in which Ethereum's accounts are abstracted as vertices of a attributed graph, while edges are used to de-note transactions. Depending on the purpose of the model, these edges can be weighted by the volume or frequency of transactions, or even a combination of both. Once this graph is constructed and populated with the raw data, it is processed using graph embedding algorithm whose main function is to map each vertex into a low-dimensional space in such a way that makes similar vertices have relevant mappings. These mappings are then manipulated by a deep learning algorithm to derive different variants of the original graph. These variants form the baseline of the normal operational regions and are therefore used to assess anomalies. The model presented in [19] is one such work that was developed based on the above approach. It uses the One Class Graph Neural Network as the graph embedding algorithm and Graph Convolutional Networks to generate the variants of the inputted graph. The dataset used to validate the proposed model was obtained by extracting all transactions that took place over the Ethereum Mainnet network from 2016 to 2017. These records are then processed manually to separate the transactions that cause or are affected by attacks from the rest using 34 features, out of 7.3 million transactions 50,422 transactions are used to assess the model. The performance readings reported by the authors underline the advantages of their proposal over the One class SVM (OCSVM) and Isolation Forest (IForest) models, however the highest accuracy reading was given as 90.75. The model of [20] follows the same approach as [19] to detect the phishing nodes. This model uses message passing as a graph embedding algorithm and uses a dataset with 100,000 la-belled accounts and 250,000 transactions to train and test the model. The results reported in this paper show the model's ability to detect phishing nodes with an overall precision of 0.935, a recall of 0.904 and an F-score of 0.919. However, the authors of this work point out some limitations, such as the inability of the static graph model to thoroughly account for the dynamic nature of transactions and the need to use a multi-attribute graph to improve prediction performance.

III. Overview of Ethereum Transaction Fees Mechanism and Dataset

The idea of blockchain is dated back to 1991 when authors of [21] were looking for as a new technology by which distributed documents can be handled securely and identified chronologically. They proposed to use a one-way hash function to morph the original documents into a cryptographic format so that the content of the documents remains secure, and a trusted time-stamping service to certify the linked records so that the creation of documents can be tracked historically. A year later, the authors of [22] propose using Merkle trees as a means of grouping multiple documents into a block and linking them in a chain-like fashion. In this way, a new block can contain a history of the entire chain.

In the last decade, a proposal to use the blockchain concept as an infrastructure for building distributed monetary systems has appeared in [2]. This work introduces a new form of currency called Bitcoin, which can be coined and traded without the intervention of a central authority. Bitcoin assures transparency as every user of this system keeps a copy of all transactions that have taken place on the Bitcoin blockchain and synchronises it whenever a new change. Furthermore, the validity of the data is enforced by a mechanism that allows the parties to agree and verify the content of the blocks (called consensus) before they are registered on the blockchain. The integrity of transactions is ensured by the fact that it is almost impossible to change the content of a block once it has been attached to the blockchain. In addition, Bitcoin uses various cryptographic primitives such as asymmetric hash functions, symmetric cryptography, digital signatures and public key cryptography to ensure confidentiality and authentication. Besides the aforementioned, Bitcoin accounts for the stability and attractiveness of its system by offering an incentive system whereby participants who contribute to the expansion of the blockchain (so-called miners) receive financial compensation for the use of their computing resources.

The success of Bitcoin in decentralising money and payment spark attention towards the enormous benefits that can be reaped from adopting blockchain in various sectors. As a result, countless new blockchain platforms have been introduced, some of which aim to improve various aspects of Bitcoin, such as Monero, which was developed to allow users to hide their private transactions, or Dash, which aims to minimise transaction costs. Other innovations aimed to add further functionalities to Bitcoin. Ethereum is the first blockchain platform that aims to allow users to develop decentralised applications (DApps) over blockchain. Ethereum achieves this goal by equipping its blockchain with the Ethereum Virtual Machine (EVM), which is able to store, validate and execute programming tasks, just as Bitcoin does with monetary transactions. Some references interpret Ethereum as the second generation of blockchain, i.e. blockchain version 2, while others see it as a complete ecosystem through which both tangible and intangible assets can be represented as smart properties.

Ethereum's founder has provided a comprehensive formal description of its internal operations and key concepts underlying it in the Yellow Paper[3] and several monographs have devoted their entire contents to describing its internal structures and how to develop applications on it. In the remaining of this section, an overview of for the mechanism that are most relevant to this work is provided and then the dataset used in this study is described.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 8, August 2022

DOI: 10.17148/IJARCCE.2022.11823

A. Overview of Ethereum Transaction Fees Mechanism

A transaction is the atomic data structure of the Ethereum blockchain which is a sequence of instructions that can be executed by the EVM, such as depositing/crediting ETH (the Ethereum cryptocurrency) to a beneficiary or deploying/calling a code (smart contract). Each transaction is identified by several fields, including, among others: (i) a counter for the number of transactions carried out by the user, (ii) the recipient, which is used to identify the address of the account for which the transaction is intended. It is worth noting that Ethereum considers the reusability of smart contract codes by regarding each smart contract as an account, called a contract account(CA) and assigning it a unique address in a different way than accounts managed by human and controlled by private keys ,called External Owned Accounts (EOA) obtain their addresses[22]. (iii) Gas information, which specifies the fees required to execute instructions on the EVM (more on this shortly). (iv) The value field contains the amount the sender wishes to pay in this transaction, hence if this transaction is addressed to a CA, the value of this field is set to zero. (v) The data field contains the information required for the execution of the smart contract and 0 if the transaction is not destined to CA. Once a transaction is initialised, it is signed with the originator's private key and then broadcasted to the peer nodes. Ethereum specifies a special protocol called Recursive Length Prefix (RLP) to ensure compact encoding and consistent serialisation of transactions across the network. When this transaction is received by a peer nodes, its verifies it from various aspects, such as whether the creator's signature is valid, whether the transaction size is less than 32 KB, which serves to mitigate DoS attacks, and whether the creator has sufficient fees to use the EVM. If the transaction meets these intrinsic verification criteria, it is propagated further to other nodes and when transaction is received by a miner, it is added to the node's mempool. This making the transaction publicly visible even though it is not yet attached to the blockchain.

Before describing the mechanism by which the transaction is added to the blockchain (the so-called mining mechanism), it is important to introduce the concept of gas. Gas is a dimensionless metric that quantifies the computational effort imposed on EVM to execute a given operation. The yellow page specifies the gas requirements for all possible Ethereum-defined low-level operations (i.e., opcodes) in such a way that reflects their complexity. For example, the integer addition costs 3 gas units, while modulo addition costs 8 gas units. It is worth noting that opcodes prices play an important role in securing Ethereum. Therefore, they are constantly reviewed and updated whenever a security vulnerability is discovered; [23] provides an updated list of opcodes and their corresponding gas prices. An example of exploiting the underpricing of some opcodes to carry out an attack was in 2016, when the creation of the blockchain was reduced by 2-3 times and causes hard-fork because EXTCODESIZE opcode (the code that reads the information of the contracts from storage) was repeatedly called. Subsequently, the Ethereum Improvement Proposal-150 (EIP) was ratified to increase the price of EXTCODESIZE by 35 times to 700 gas units. Theoretically, the total amount of gas units required to execute a given transaction can be calculated as the sum of the gas units of its opcodes. While this is true for some classes of simple transactions (e.g., moving some ETHs between accounts or performing some arithmetic operations), applying this calculation approach to more sophisticated transactions (e.g., retrieving EVM environmental information or deploying a new smart contract) does not always result in accurate gas units. Several papers investigating gas characteristics of transactions e.g.,[24] have suggested some reasons for this inaccuracy such as throwing an undecidable halting exception during the execution of some opcodes, the discrepancy amongst miners' EVM versions and the changes in blockchain states during the execution of transactions. Interestingly, Ethereum attempts to mitigate the negative impact of miscomputation of gas units by letting a transactor to set the maximum amount of gas units they are willing to spend to execute their transaction in the gasLimit field.

In the legacy version of Ethereum, i.e., before the enforcement of EIP-1599, the originator of the transaction must also specify the price she/he is willing to pay for each unit of gas in the gasPrice field of the corresponding transaction. The value of gasPrice is defined in gwei, where $1\ gwei = 10^{-9}\ ETH$. Hence, multiplying gasPrice by gasLimit yields the amount of gewi that the creator of a transaction is willing to pay to execute thier transaction. The importance of setting an appropriate gas price stems from the fact that Ethereum's incentive mechanism relies heavily on this value. More specifically, the reward that miners receive for including and mining a transaction in the next block is the sum of the gas units used to execute each opcode, charged at the gasPrice (also called the transaction fee). Although Ethereum allows each miner to set a lower limit on the value of gasPrice, beyond/below which a received transaction can be ignored, there is no constraint on the upper limit of gasPrice. This approach sounds good in that it allows a transactor to prioritise its transactions by setting a high gasPrice for transactions that are subject to fierce competition (e.g. participating in a Metaverse property auction or deploying new NFTs during network congestion). However, the absence of a cap on gasPrice has led to several monetary losses and attacks. Omnipresent sandwich attack[25] is one such attack that exploits the relationships between the gasPrice and the transaction order to sandwich the victim transaction between front-running and back-running fake transactions that works to seize the trading opportunities by manipulating of the victim transactions' execution orders.

It is worth to note that since the London Hard fork an improvement prosal EIP-1599 was enacted in August 2021. This proposal not only changes the way a transaction fee is specified but also the beneficiaries to whom it goes. The new transaction fee is divided into two parts: Base Fee per gas unit ($baseFeePerGas$) and Tip Fee per gas unit ($PriorityFeePerGas$). The former is defined as the minimum price per unit of gas that is required to be paid for each transaction to be included in the new block. The base fee is controlled by the protocol itself to reflect network usage, and it is burnt by sending it to a verifiably un-spendable address. The tip, on the other hand, is set by the originators of transactions to control the execution priority of their transactions and paid to the miner. In addition, EIP-1599 allows the

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 8, August 2022

DOI: 10.17148/IJARCCE.2022.11823

user to specify the maximum amount of fees per unit of gas that they are willing to pay ($maxFeePerGas$). This value serves as the fee cap for the transaction.

Another radical change introduced by EIP-1599 is the use of a variable block size instead of the fixed block size used in the legacy version. More specifically, prior to EIP-1599, the Ethereum protocol required that each block not exceed a certain amount of gas units ($BlockMax$). This was 12.5M+0.0976% before the Berlin Hard Fork in April 2021 and 15M gas units thereafter. The main purpose of setting $BlockMax$ was to prevent the network from being overloaded with very computationally intensive blocks that could otherwise increase propagation delay. However, when transaction demand is high, imposing such limit can delay the inclusion of transactions to the block, impoverish the user experience and increase the possibility of bribery attacks. The EIP-1599 replaces this static threshold with the definition of a dynamic target value ($BlockTarget$, currently set at 15 million gas units) and allows the value of BlockTarget to be increased or decreased it in consistency with the number of outstanding transactions, provided that the maximum amount of gas units in each block ($BlockMax$) is less than or equal to 2*$BlockTarget$.

Based on the above the above, the base fees of a block can be computed as:

$$BF_t := BF_{t-1}\left(1 + \frac{1}{8}\frac{S_{t-1} - S_{target}}{S_{target}}\right) \quad (1)$$

where BF_t is base fee of the block at time instance t, BF_{t-1} is base fee of the predecessor block, S_{target} and S_{t-1} are the size of the target and predecessor blocks respectively. The main principle sought by the equation (1) is to stabilise the fluctuations of the base fee in a suitable ratio (in this case, one-eighth). For example, if there are lots of outstanding transaction then the S_{t-1} will be set to $2 \times S_{target}$ which in turn leads to increase the base fee of the new block a factor of 12.5%, on the other hand, under idle demand, i.e. if st-1 is 0, then the base fee will be reduced by the same percentage. Table I summarizes the main differences between the gas-related parameters before and after enforcing EIP-1599.

TABLE I Transactions Fees parameters before and after EIP-1599

Parameters	Before EIP-1559	After EIP-1559
Mechanism	First-price auction	posted-price mechanism
Block size policy	fixed-sized blocks	Variable sized blocks.
$BlockTarget$ (gas unit)	-	15 million
$BlockMax$ (gas unit)	15 M	$2 \times BlockTarget$
$gasPrice$	The amount of gewi per gas unit that user is committed to pay in order to execute transaction.	-
$gasLimit$	Maximum amount of gas units a transactor is willing to spend to execute their transaction.	-
$gasUsed$	The actual amount of gas unit the is used to execute the transaction on EVM.	Same as before.
$baseFeePerGas$	Not defined	The minimum amount of gwei per gas unit required to include the transaction in the block(computed algorithmically as in equation XX)
$PriorityFeePerGas$	Not defined	The amount of gwei per gas unit that the transactor pays to miner directly as incentive.
$maxFeePerGas$	Not defined	The maximum gwei per gas unit that a user is willing to pay.
Amount of gas paid to miners (gewi)	$gasPrice \times gasUsed$	$gasUsed \times PriorityFeePerGas$
Amount of gas burned (gewi)	Not applicable	$gasUsed \times baseFeePerGas$
Total amount of gewi paid by transactor to execute transaction	$gasPrice \times gasUsed$	$gasUsed \times \min(maxFeePerGas, baseFeePer + PriorityFeePerGas)$

B. Dataset

To evaluate the performance of the proposed model on a real dataset, all transactions of Ethereum Minnet since 1/6/2016 were extracted. Each transaction is characterised by the following attributes: *Transaction hash*, which serves as a unique identifier. *Status*, which describes the current situation of the transaction. This attribute can take the following values: success, pending or failed, corresponding to the following cases: when the transaction has been successfully executed and added to a block, when it is waiting to be executed and finally when the execution of the transaction results

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 8, August 2022

DOI: 10.17148/IJARCCE.2022.11823

in errors respectively. *Block* is the block number comprising the transaction, the timestamp at which the block was mined, *From* and *To* attributes are used to describe the originator and recipient of the transaction respectively. Besides the above these attributes contains all the above parameters that are used to signify the transaction fees. Figure 1 illustrates the correlation matrix of these attributes whereas table II summarises the main types of attacks that have been extracted from these dataset along with their number of transactions.

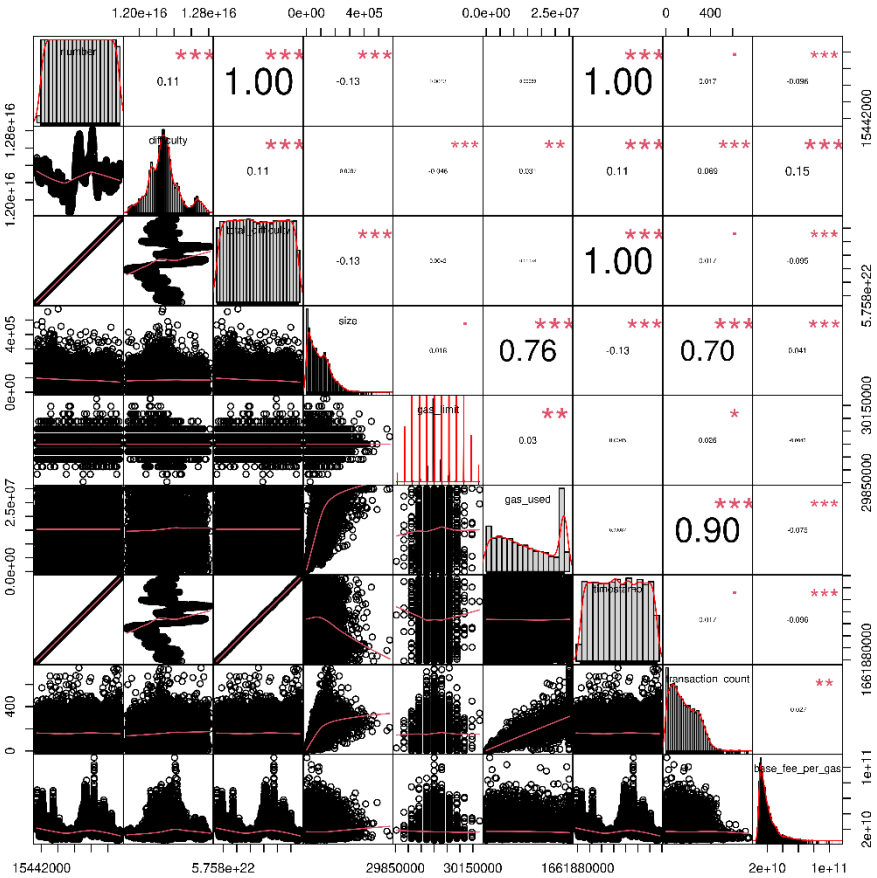

Fig. 1 Corrleation Matrix of the transaction's attributes

TABLE II Number of transactions of different types of attacks

Attack name	**Description**	**Number of transactions**
front-running	Snip the profitable transactions, copy them but with higher gas prices, and send the copied versions to miners.	112541
Sandwich	Same as above, but sandwich the victim transactions are sandwiched between front-running and back-running malicious transactions.	125415
Block Stuffing	Fill up the block with many transactions to prevent others from inclusion their transactions.	325141
Time-Bandit	Rewrite the blockchain history to steal funds from previous transactions	54142
Transactions manipulating	Change the transaction orders or inclusion either to get self-benefits or for others in front of bribing.	6984471
fee-based forking attack	Tamper the blockchain by conducting a hard fork to make more profits.	11428562
Soft forking	Repeat the mining process of the previous blocks with the aim to duplicate the rewarding and then fork the block chain to hide the duplicated blocks.	2212211
Normal	Not falling within the above attacks	112297112
Total		133539595

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 8, August 2022

DOI: 10.17148/IJARCCE.2022.11823

IV. THE PROPOSED MODEL

In cases where CNN models are designed to process image data, many similar layers (e.g. Inception Unit in GoogLeNet and Residual Block in ResNet) are placed in the model and then one or more pruning techniques (e.g., dropping out some layers or skipping some connections of a layer as in ResNet) are applied when overfitting occurs. This design approach is based on the assumption that instances of image data are hierarchically contextualised. Therefore, running multiple identical layers can gradually reveal these features. More specifically, most current CNN models are constructed from multiple identical tuples of "convolutional layers, non-linear layers and pooling layers". Typically, the earlier tuples are dedicated to extract the primitive features of objects in the image, such as object edges and contours. In contrast, the later layers are used to extract more latent features within discovered objects. However, the significant differences between the transaction dataset considered in this study and the image dataset make it difficult to use the same design approach. Therefore, we propose to use more specific layers instead of general identical layers. By this means, we can use the layer that is appropriate for identifying the features we are looking for instead of blindly adding many identical layers.

Based on the fact that a transaction is a generic type of time series, typically composed of different patterns, and that decomposing the transaction into elementary patterns greatly facilitates its interpretation. This in turn leads to an improvement in predictive performance, we use time series decomposition approaches [26] as a framework for developing our task-specific layers. Time series decomposition approaches are among the well-established statistical methods used to analyse time series in various disciplines; they substantiates that an instance of a time series can be projected onto two fundamental components: the seasonal component and the trend. The seasonal component is used to describe the behaviour of a time series that recurs at regular intervals, while the trend component quantifies the persistence behaviour that the time series exhibits. Correspondingly, we propose two novel layers: the seasonal detecting and the trend-discovering layers. The following subsections describe the design philosophy of these two layers, while subsection xx presents the overall model.

A. Seasonal-Detecting Layer

The recent studies devoted to analysing and modelling Blockchain transactions, e.g., [27] reveal that most of them possess some sort of seasonality, ranging from a naïve short-term correlation up to more sophisticated long-term dependencies and/or fractals. Therefore, several hypotheses have been put forward to reveal the root causes of such a phenomenon. Some works claim that seasonality is a natural result of several monetary phenomena that dominating the cryptocurrency markets such as crowdfunding or occurrences of NFT drops. The mainstream technique that is usually used to detect the seasonality of a time series is to divide it into several subintervals and then apply a similarity metric to quantify to which extent an interval is rhythmic to other(s); autocorrelation function, chi-square periodograms, Hurst exponent and Durbin–Watson statistic are some examples of such technique. Nevertheless, the primary aim here is neither to identify whether or not a given time series possesses some degree of seasonality nor to quantify the seasonality degree. Rather, our aim is to design a CNN-compatible layer that can extract the seasonal features presented in the intended traffic and then encode these features into the model's parameter space. Thus the model can take seasonal characteristics into account when making a prediction for future readings. To achieve this aim, we employ dilated convolutional in the lieu of standard convolution; in dilated convolution, one or more zero-valued cells interleave(s) the kernel's adjacent cells at regular intervals; this yields a larger yet hollow kernel. Applying such kernel over a time series enables each receptive field of the resultant features map to capture several behaviours of the traffic at non-consecutive temporal steps. Inasmuch as these features maps are part of the hypothesis space of the model, the relations that might be existing within the receptive fields of the same features maps or even across several features maps can be revealed as the learning proceeds, which in turn leads to encoding the seasonality features onto the model's parameters. Figure 2 provides a schematic diagram for a dilated convolution that takes place between time series comprises T time steps and a kernel of length 3 with dilation rate, denoted here by $F_{R_k}^{(l,k)}$, of 1.

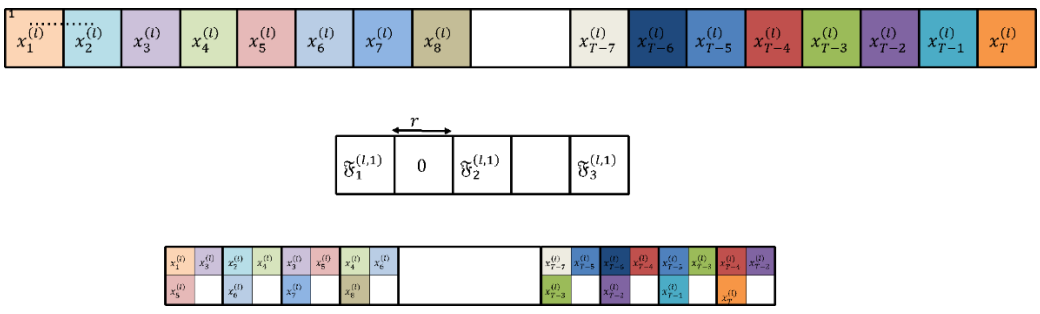

Fig. 2 A schematic diagram for seasonal-detecting layer

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 8, August 2022

DOI: 10.17148/IJARCCE.2022.11823

As it can be seen from this figure, each receptive field of the feature map is generated from processing three input elements (the width of the kernel) that are separated by one (the dilation rate); for instance, the input elements used to generate the receptive fields $y(t)$ is $x_t^{(l)}$, $x_{t+2}^{(l)}$and $x_{t+4}^{(l)}$ or in general form: $x^{(l)}_{t+n\left(F_{R_k}^{(l,k)}\left(F_{D_k}^{(l,k)}-1\right)\right)}$; $0 \le n \le F_{D_k}^{(l,k)} - 1$.

Our aim here is to set the dilated convolution's parameters (i.e., the width of kernel and dilation rate) to those values that allow the resultant features map to possess all possible seasonal features presented in the given input. This aim can be achieved readily if $F_{R_k}^{(l,k)}\left(F_{D_k}^{(l,k)} - 1\right)$ is set to the fundamental period of the given input, i.e., the fundamental period is the smallest interval at which the seasonal behaviour is repeated. Precisely, this setting leads to accommodate n identical instances of the time series into each receptive field which, in turn, yields a features map with a broad view that covers the global seasonal behaviour presented in the time series. Interestingly, in this way the resulting feature map can be conceptualised as a long-term memory unit of equidistant input instances and the proposed seasonal-detecting layer as the foremost convolutional layers that are typically used in the CNN image-based models to detect information about contours and edges of objects within an image.

B. Trend-discovering Layer

The trend-discovering is the second specific-task layer that is proposed to improve the performance of the CNN model by allowing it to use a more specialised layer instead of nesting multiple layers of general structure as the case in conventional CNN models. The principal approach of this layer is to capture the persistence behaviour exhibited in transactions and then to encode them into the hypothesis space of the model in a similar way that the seasonal-detecting layer does for the seasonal characteristics. It is interesting to investigate the main causes of the trends in an transaction streaming and the importance of discovering them in improving the prediction performance of the CNN before discussing the design methodology used to devise the specific-task layer. Recalling that the transactions is not of deterministic nature; rather, it varies due to the interaction of several stochastic processes, e.g., volatility of market and redistribution of tokens. While some of these changes persist for a long interval (we refer to them here as here trend), others elapse for a lesser interval or even leave off soon (i.e., dubbed transient state). Hence, by discovering the persistence (trends) of the traffic, the CNN can avoid making a wrong prediction based on a transient state, which can leverage the CNN model's predictability.

Most of the methods that have been proposed to discover the trend of a time series fall into two groups: parametric and nonparametric. The former is based on the assumption that the trend of the series follows a particular function. e.g., linear, quadratic and cubic; therefore, by computing the function's parameters from the dataset, the trend characteristics of the time series can be determined. Conversely, the nonparametric approaches impose no presumptions on the shape or form of the trends functions; rather, it filters out the striking variations by applying smoothing algorithms of a predefined window over the time series. Several smoothing algorithms, including unweighted sliding-average smooth [], triangular smooth and Savitzky–Golay filter with or without predefined weights such as Binomial weights, spencer's 15 points moving average, have been proposed to deal with trends. Considering the fact that most of the smoothing algorithms are based on the convolution operation facilitates (using standard convolutional layer as a trend trend-discovering layer) building a CNN compatible trend-discovering layer.

C. Overall CNN models

Figure 3 shows the block diagram of the proposed overall model. It can be seen that this model consists of three branches, each of which has roughly the same structure. The first branch consists of a seasonal detection layer with 500 filters and a dilation rate of 365, followed by a non-linearity layer with a ReLU activation function whose objective is to smooth the feature map generated by the seasonal detection layer. The output of the first layer is a dropout layer with a probability of 0.1, which is used to eliminate the possibility of overfitting, and finally a max-pool with a width of 2, which is used as a dimension reduction algorithm. The number of layers was chosen to provide a sufficient number of filters to find out all possible combinations of latent patterns in the given transaction sequence. Interestingly, the dilation rate was set to 365 as this represents the number of days in each year. This in turn facilitates the detection of the seasonal behaviour exhibited by the input data on each day of the year. The output of the Max Pool layer is then passed to the trend-discovering layer, whose main objective is to determine the trends of each sub-interval generated by the seasonal detection layer. The seasonal-detecting layer of the second branch is designed in such a way that allows it to detect the seasonal features on a monthly basis. Therefore, the dilation rate of 30 is used instead of 365 as in the first branch; similarly, the rate of 1 is used in the seasonal detection of the last branch to find out the seasonal features on daily basis. The output of the three branches is passed to the flatten layer, whose main function is to convert the multidimensional data generated by all layers into a sequence before concatenating them together. The results of this last layer are then processed by two fully concatenated layers using the softmax activation function to generate the appropriate class for each input instance.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 8, August 2022

DOI: 10.17148/IJARCCE.2022.11823

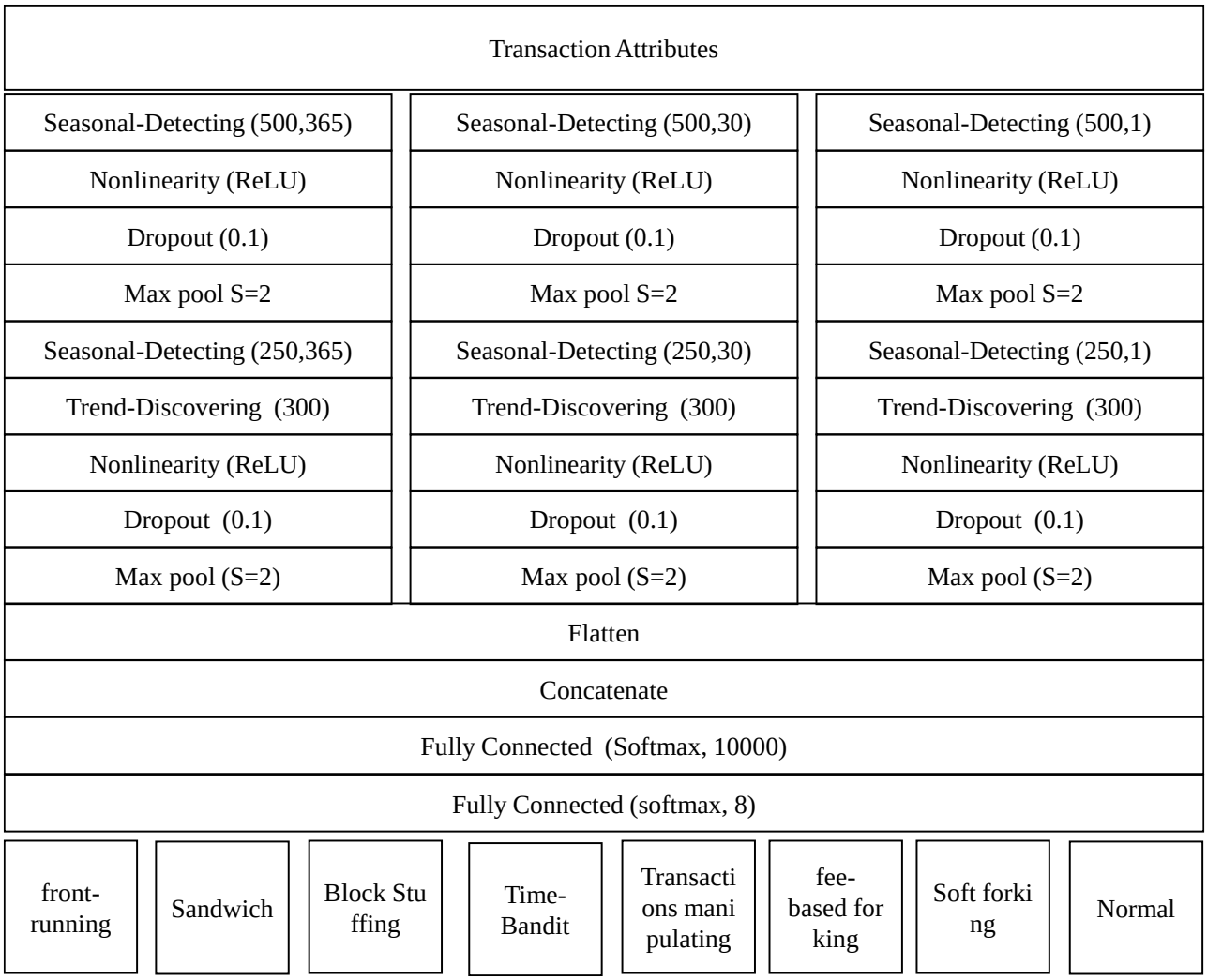

Fig. 3 Block diagram of the proposed model

V. RESULTS AND DISCUSSION

A. Performance Metrics

The accuracy of the proposed model is evaluated by comparing its prediction results with the ground truth data obtained from the three datasets described above. The following statistical measures are used in this evaluation: Accuracy, Precision, recall, F1Score. The definitions of these metrics are given using the following terms: True Positive (TP), False Positive (FP), True Negative (TN) and finally False Negative (FN). TP is defined as the total number of transactions classified by our model as being within a given class in accordance with the ground truth; TN, on the other hand, has the same interpretation as TP, except that it groups together the transactions that fall outside a given class. TN counts the total number of transactions that are classified by the proposed model as belonging to a certain class, while this is not the case in the ground truth. FN finally, is used to describe the total number of transactions that are classified by the model as belonging to a class, while in reality they are not in that class. Accordingly, the Accuracy, Precision, recall, F1Score can be defined mathematically as following:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

$$\text{precision} = \frac{TP}{TP + FP} \tag{3}$$

$$\text{recall} = \frac{TP}{TP + FN} \tag{4}$$

$$\text{F1} - \text{score} = \frac{2TP}{2TP + FP + FN} \tag{5}$$

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 8, August 2022

DOI: 10.17148/IJARCCE.2022.11823

This study uses the latest version of the Python programming language, namely 3.10.6, to implement the proposed models as well as other benchmarking models. The Tensorflow library with Keras was chosen for its wide applicability in this field. All codes are deployed on a single machine with an Intel Core i7-11850HE processor running at 4.70 GHz with 8 cores and 128 GB RAM and an AD102 GPU (RTX 4090) GPU. The Adam algorithm is used as an optimiser for the proposed model with $\beta_1 = 0.91$ and $\beta_2 = 0.99$ momentum respectively, the number of epochs is set to 150, the batch size is set to 1024 instances and the learning rate is set to 0.0001.

This study uses the 6-fold cross-validation strategy, in which each of the datasets considered in this study is randomly divided into 6 sub-datasets, 5 of which are randomly selected and then used to train the model, while the remaining sub-datasets are used to validate the accuracy of the proposed model.

The first evaluation of the proposed model is conducted by measuring how the normalised errors and the accuracy of the model change as a function of the number of epochs. The results of this evaluation are shown in the figure 4. From these two figures, it can be seen that the normalised error decreases exponentially with the number of epochs. This is due to the ability of the proposed model to acquire more knowledge over time and to anchor this knowledge firmly in its own parameter spaces. In particular, the use of multiple seasonal-detecting layers to recognise seasons facilitates the detection of patterns that repeat at different intervals. This in turn converts the long series of transactions into smaller intervals so that the model can easily capture the recurring patterns. In addition, by using the trend-discovering layers, the model can find out the characteristics of the underlying process that dominates the generation of the transaction. The combination of these two task-specific layers gives the model a powerful tool to interpret the various features of the transaction dataset without the need for significant learning effort. It is shown, for example, that the model reaches the zero-error range in less than one tenth of the predefined epochs. The normalised accuracy curves as shown in figure xxx demonstrates that the accuracy of the model increases as soon as the normalised error drops.

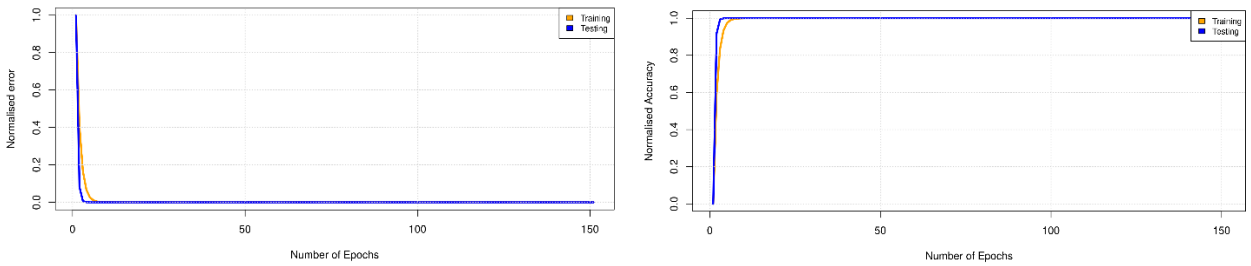

Fig. 4 Normalised error and accuracy of the proposed model during training and testing phases

From the confusion matrix shown in Figure 5, it is clear that the model can classify different types of attacks occurring in different transaction instances with a high degree of accuracy. These characteristics support the rapid learning curve discussed earlier, as you can see that the lowest value 0.98 corresponds to the sandwich attack. Should recalling that in this attack, requires to proceed and follow a victim transaction by two malicious transactions whose gas prices are lower and higher than the victim's gas prices, respectively. This in turn adds some challenges in figuring out the pairing of these transactions before deciding whether they belong to this type of attack or not. We found that the sandwich attack requires padding the block with many transactions to eliminate the possibility of others benefiting from the same opportunity seized by the attackers, hence some cases of sandwich attacks can be seen to have leaked to block stuffing attack. The second lowest value in this confusion matrix relates to the front-running attack, which is due to the fact that this type of attack is fast, i.e. as soon as the snipper bots realise that there is a chance to gain advantage, they are replicated with a higher gas price and sent back to the same mempool. Also, since most of these bots run publicly, it is possible that they will be repeated by other bots that snipe the last snipper, which in turn leads to a long tail attack and requires more effort to fully detect them. In addition, it can be seen that other types of attacks can also be classified at a rate of 1 true positive, as each of these attack types has its own characteristic patterns that can be easily identified.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 8, August 2022

DOI: 10.17148/IJARCCE.2022.11823

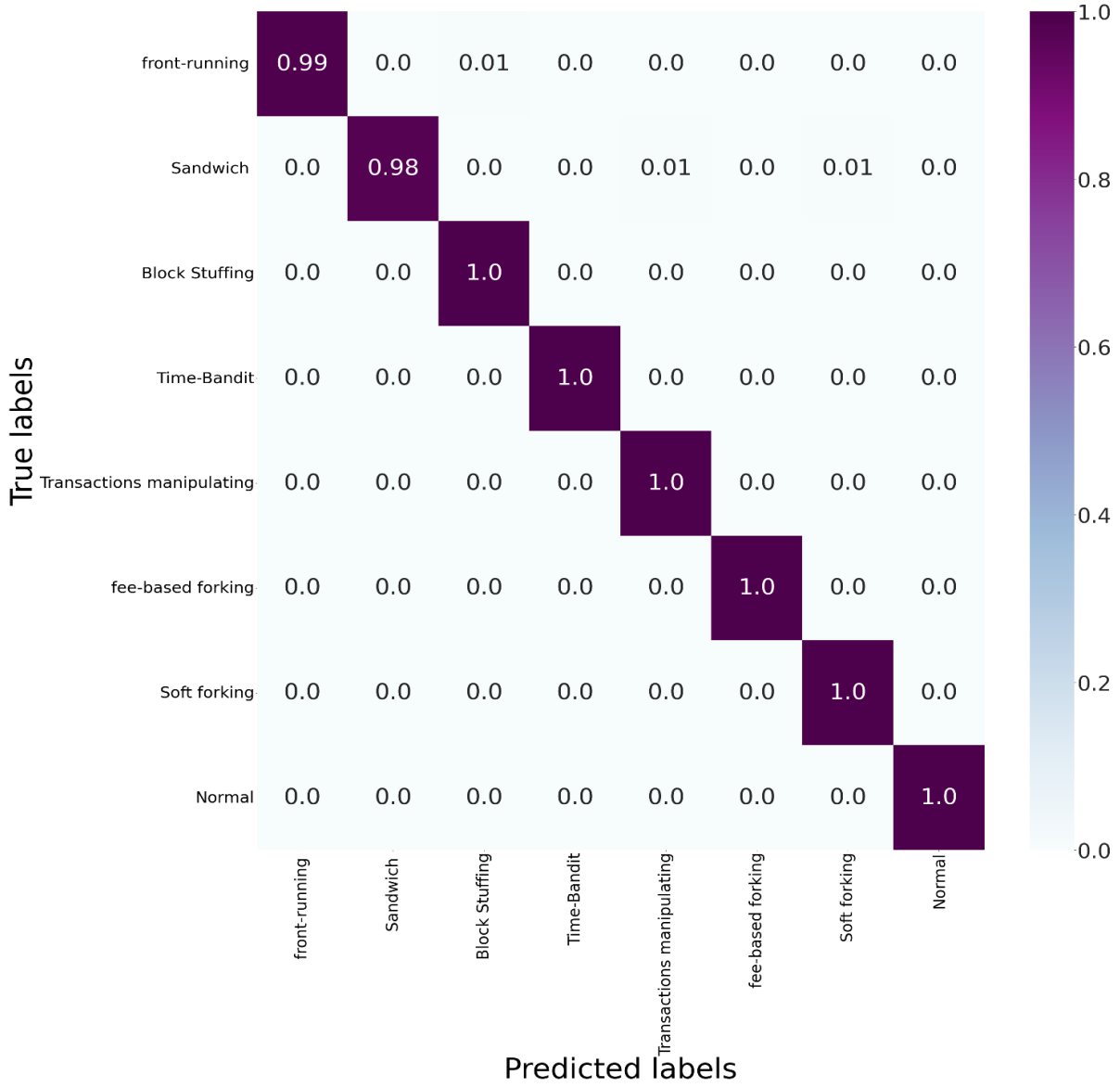

Fig. 5 Block diagram of the proposed model

Table III summarises the model's performance metrics recorded during the different training/testing folds. From these results, it is clear that the model is robust in terms of the size of the training and testing data. This is important because it allows the model to be used in different situations without having to be trained over a long period of time. These results also highlight the crucial advantages that the specific task layers offer in classifying the different types of attacks on the Ethereum blockchain.

TABLE III Performance metrics of the proposed model during different training-testing folds

Number of folds	Accuracy	Precision	Recall	F1Score
1	99.97	100.0	100.0	99.99
2	99.99	99.99	100.0	100.0
3	100.0	100.0	99.99	99.99
4	99.99	99.98	100.0	100.0
5	100.0	100.0	100.0	99.97
6	100.0	100.0	100.0	99.99

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 8, August 2022
DOI: 10.17148/IJARCCE.2022.11823

VI. CONCLUSION

In this paper, we propose a novel temporal convolutional neural network that can be used to detect malicious transactions in one of the most important blockchain infrastructures, namely Ethereum. The proposed model was developed out of the need to protect this infrastructure from the various types of attacks that target it and lead to catastrophic consequences. The underlying concept of the proposed model is to treat a transaction as a stochastic time series and then develop two specific task layers that are compatible with the traditional CNN architecture. The first layer is responsible for detecting the seasonal characteristics of the transactions, while the second layer is used for detecting the trend. These two layers are integrated with the traditional architecture to form a powerful temporal CNN architecture that can classify different types of attacks. The performance of the proposed model was evaluated from a different perspective by using real transactions from the Ethereum Main-Net network. The results of the comprehensive evaluations show the ability of the proposed model to perfectly identify malicious transactions in the Ethereum blockchain.

REFERENCES

[1]. Li, Kuan-Ching, et al., eds. Essentials of blockchain technology. CRC Press, 2019.
[2]. Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." Decentralized Business Review,2008.
[3]. Wood, Gavin. "Ethereum: A secure decentralised generalised transaction ledger." Ethereum project yellow paper 151.2014,2014.
[4]. Goodman, L. M. Tezos: A self-amending crypto-ledger position paper. Aug, 3, 2014.
[5]. Bahga, Arshdeep, and Vijay Madisetti. Blockchain applications: a hands-on approach. Arshdeep Bahga & Vijay Madisetti., 2017.
[6]. Torres, Christof Ferreira, and Ramiro Camino. "Frontrunner jones and the raiders of the dark forest: An empirical study of frontrunning on the ethereum blockchain." 30th USENIX Security Symposium (USENIX Security 21). 2021.
[7]. Lewis, Michael. Flash boys: a Wall Street revolt. WW Norton & Company, 2014.
[8]. Fang, Lucius, et al. How to DeFi: Advanced. Vol. 1. CoinGecko, 2021.
[9]. MEV-Explore, https://explore.flashbots.net/ online access on (15/8/2022).
[10]. Maximum Extractable value, https://ethereum.org/en/developers/docs/mev/ online access on (15/8/2022).
[11]. Wen, Yujuan, et al. "Attacks and countermeasures on blockchains: A survey from layering perspective." Computer Networks 191,2021.
[12]. Maleh, Yassine, et al., eds. "Blockchain for cybersecurity and privacy: architectures, challenges, and applications.", 2020.
[13]. Aggarwal, C.C. Neural Networks and Deep Learning: A Textbook, 1st ed.; Springer Nature: Cham, Switzerland, 2018.
[14]. Torres, Christof Ferreira, and Ramiro Camino. "Frontrunner jones and the raiders of the dark forest: An empirical study of frontrunning on the ethereum blockchain." 30th USENIX Security Symposium (USENIX Security 21). 2021.
[15]. Varun, Maddipati, Balaji Palanisamy, and Shamik Sural. "Mitigating Frontrunning Attacks in Ethereum." Proceedings of the Fourth ACM International Symposium on Blockchain and Secure Critical Infrastructure. 2022.
[16]. Scicchitano, Francesco, et al. "A Deep Learning Approach for Detecting Security Attacks on Blockchain." ITASEC. 2020.
[17]. Scicchitano, Francesco, et al. "Deep autoencoder ensembles for anomaly detection on blockchain." International Symposium on Methodologies for Intelligent Systems. Springer, Cham, 2020.
[18]. Praitheeshan, Purathani, et al. "Security analysis methods on ethereum smart contract vulnerabilities: a survey." arXiv preprint arXiv:1908.08605,2019.
[19]. Patel, Vatsal, Lei Pan, and Sutharshan Rajasegarar. "Graph deep learning based anomaly detection in ethereum blockchain network." International Conference on Network and System Security. Springer, Cham, 2020.
[20]. Yu, Tong, et al. "MP-GCN: A Phishing Nodes Detection Approach via Graph Convolution Network for Ethereum." Applied Sciences 12.14: 7294, 2022.
[21]. Haber, Stuart, and W. Scott Stornetta. "How to time-stamp a digital document." Conference on the Theory and Application of Cryptography. Springer, Berlin, Heidelberg, 1990.
[22]. Chittoda, Jitendra. Mastering Blockchain Programming with Solidity: Write production-ready smart contracts for Ethereum blockchain with Solidity. Packt Publishing Ltd, 2019.
[23]. EVM OPCODE Gas Costs, https://github.com/djrtwo/evm-opcode-gas-costs online access on (15/8/2022).
[24]. Yang, Renlord, et al. "Empirically analyzing ethereum's gas mechanism." 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). IEEE, 2019.
[25]. Heimbach, Lioba, and Roger Wattenhofer. "Eliminating Sandwich Attacks with the Help of Game Theory." Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security. 2022.

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 8, August 2022
DOI: 10.17148/IJARCCE.2022.11823

[26]. Hyndman, Rob J., and George Athanasopoulos. Forecasting: principles and practice. OTexts, 2018.
[27]. Liu, X. F., Akcora, C. G., Zhang, Z. Y., & Liu, J. G. (2022). Cryptocurrency Transaction Analysis From a Network Perspective. Frontiers in Physics, 215.

BIOGRAPHY

Mohammed Baz (Senior Member, IEEE) is currently an Associate Professor at the Department of Computer Engineering, University of Computer and Information Technology, Taif University, Taif, Saudi Arabia. Baz received his PhD from the University of York on applications of statistical inference in the design of communication protocols for low power wireless networks. He has published a number of papers at recognised conferences and has served as a reviewer for a number of IEEE journals, including IEEE Transactions on Vehicular Technology, IEEE Access journal and IEEE Wireless Communications Letters. He has been a member of several committees in academic areas and has participated in research projects. He has also taught several courses and supervised several undergraduate and graduate student capstone projects.