Collaborative Learning Framework to Detect Attacks in Transactions and Smart Contracts

Tran Viet Khoa, Do Hai Son, Chi-Hieu Nguyen, Dinh Thai Hoang, Diep N. Nguyen, Tran Thi Thuy Quynh, Trong-Minh Hoang, Nguyen Viet Ha, Eryk Dutkiewicz, Mohammad Abu Alsheikh, and Nguyen Linh Trung.

***Abstract*—This paper presents a novel collaborative learning framework designed to detect attacks in blockchain transactions and smart contracts by analyzing transaction features. The proposed framework incorporates a unique tool that transforms transaction features into visual representations, facilitating efficient analysis and classification of low-level machine code for attack detection. Furthermore, we propose an advanced collaborative learning model to enable real-time detection of diverse attack types at distributed mining nodes. In order to evaluate the performance of our proposed framework, we deploy a pilot system based on a private Ethereum network and conduct multiple attack scenarios to generate a novel dataset. To the best of our knowledge, our dataset is the most comprehensive and diverse collection of transactions and smart contracts synthesized in a laboratory for cyberattack detection in blockchain systems. Our framework achieves a detection accuracy of approximately 94% in extensive simulations, which is about 22% higher than that of a centralized learning model. In real-time experiments, it achieves 91% accuracy with a throughput of over 2,150 transactions per second. These compelling results validate the efficacy of our framework and showcase its adaptability in addressing real-world cyberattack scenarios.**

***Index Terms*—Cybersecurity, cyberattack detection, deep learning, blockchain, smart contract.**

I. Introduction

A. Motivation

BLOCKCHAIN technology has been rapidly developed with many applications in recent years. This technology was initially developed with a well-known digital currency application named Bitcoin. After that, many potential applications using this technology have been developed beyond cryptocurrency. The rapid development of this technology is due to its ability to provide a new approach to data sharing and storage without the need for any third party (e.g., banks and governments). Blockchain is a decentralized environment in which transactions and smart contracts can be recorded and executed in a secure and transparent manner. It is challenging to manipulate transactions once they are put into the blocks. Thus, blockchain technology protects data integrity, and its applications have been widely developed in various fields of industry, such as smart manufacturing, supply chain management, food production, smart grids, healthcare, and the Internet of Things [1]–[4].

Smart Contracts (SCs) are solely programs in blockchain systems (e.g., Ethereum and Solana). SCs define and enforce a set of rules for users through code. They also facilitate users' interactions by allowing them to send transactions to execute defined functions. In practical scenarios, attackers can inject malicious code into SCs and transactions before deployment to attack a blockchain system for specific purposes. By default, SCs and their interactions are irreversible after deployment in a blockchain system [5]. If the attacked SCs and transactions are validated in the blockchain network, the consequences are inevitable [6]. For instance, SCs exhibit various vulnerabilities [7], which attackers can exploit to engage in injurious purposes, including unauthorized coin withdrawals from other users' accounts and taking control of the system [6], [8]–[10]. Specifically, in 2016, an SC named Decentralized Autonomous Organization (DAO) was a victim of a re-entrancy attack. At that time, this SC held $150 million in the Ethereum network, and this attack led to a hardfork of Ethereum that created Ethereum Classic (ETC) [7]. In addition, the 4Chan group created an SC named Proof of Weak Hands Coin (PoWHC) on the Ethereum system. However, this SC witnessed an underflow attack that caused a loss of 866 ETH (i.e., Ethereum coins) [11]. Although most of the attacks in blockchain systems happened in the finance sector, many blockchain-based applications have been developing in different sectors such as healthcare, supply chain, and food industry [12], [13]. Therefore, securing blockchain systems against cyber threats has become an imperative necessity.

B. Challenges in Detecting Attacks in SCs and Transactions

There are a number of challenges to detect and prevent attacks in transactions and SCs. **The first challenge is the lack of a dataset synthesized in the laboratory for various kinds of attacks on transactions and SCs in a blockchain system.** In recent research (e.g., [14] and [15]), the authors use datasets from the public blockchain network and label data using the attack records history. When using this method to label attack data, it is assumed that the benign data does not

This research was supported by the Project 02-2025-HV-VT1. This research was supported in part by the Australian Research Council under the DECRA project DE210100651. *(Corresponding author: Trong-Minh Hoang.)*

T. V. Khoa and M. A. Alsheikh are with the University of Canberra, Australia (e-mail: {khoa.tran, mohammad.abualsheikh}@canberra.edu.au).

C. H. Nguyen, D. T. Hoang, D. N. Nguyen, and E. Dutkiewicz are with the School of Electrical and Data Engineering, University of Technology Sydney, Sydney, NSW 2007, Australia (e-mail: hieu.c.nguyen@student.uts.edu.au, {hoang.dinh, diep.nguyen, eryk.dutkiewicz}@uts.edu.au).

D. H. Son is with the School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Australia and the VNU Information Technology Institute, Hanoi, Vietnam (e-mail: dohaison1998@vnu.edu.vn).

N. L. Trung, T. T. T. Quynh, and N. V. Ha are with the University of Engineering and Technology, Vietnam National University, Hanoi, Vietnam (e-mail: {linhtrung, quynhttt, hanv}@vnu.edu.vn).

Trong-Minh Hoang is with the Posts and Telecommunications Institute of Technology, Vietnam (e-mail: hoangtrongminh@ptit.edu.vn).

include attacks. Therefore, generating data, which has "clean" samples (i.e., transactions between users without any malicious behavior) of normal behavior and attacks in transactions and SCs, is urgently needed. However, a blockchain system in the mainnet has a large and diverse types of data. Thus, a synthesized dataset from the laboratory needs to be diverse and representative of real-world settings. **The second challenge is to understand and analyze the content of Bytecode, the compiled form of an SC's source code.** The core functions of transactions and SCs are encoded into the Bytecode, which is represented by a series of hexadecimal numbers, to be implemented in a blockchain system [6]. It is crucial for a real-time attack detection system to analyze the content of Bytecode to detect attacks in a blockchain system [14]. There are two approaches to analyze the Bytecode, i.e., using the source code of SCs for comparison and analyzing the Bytecode. Unfortunately, only 1% source code of SCs is open [14], and analyzing Bytecode without the corresponding source code of SCs and transactions can be unreliable and time-consuming [14]. **The third challenge is that most of the current attack detection models are centralized.** Thus, they need to gather all data (i.e., transactions together with their labels, e.g., attack or normal) into a centralized model to perform training and testing. However, blockchain systems are decentralized environments, so it is challenging to collect data from all mining nodes (MNs) to perform training at the centralized server. In addition, if we transfer data from all MNs to the centralized server for processing (e.g., training and testing), data privacy can be compromised.

C. Our Proposed Solution

Given the above, in this paper, we first set up experiments in our laboratory to deploy various kinds of attacks on transactions and SCs in a blockchain system (i.e., a private Ethereum system). These attacks were recorded as occurring in the real world and resulted in serious consequences for the Ethereum system. To address the first challenge, we collect all the transactions in MNs to build a novel dataset, called **Blockchain Transaction-based Attacks Dataset (BTAT)**[1]. To the best of our knowledge, this is the first cyberattack dataset on transactions and SCs in a blockchain network synthesized in a laboratory. To ensure diversity and realism in our dataset, we create a large number of individual accounts (i.e., 10,000 accounts) to send random transactions to the blockchain network for execution. Furthermore, to enrich the dataset, we deploy multiple SCs and apply different transaction functions to them, while maintaining consistency in the exploitation techniques used for the attacks. This dataset can be used for both research and industry purposes to address cyberattacks in transactions and SCs. In addition, to deal with the second challenge of Bytecode analysis, we propose a novel convolutional neural network-based (CNN-based) framework that analyzes transactions and SCs without the need to understand the SC source code. The main goal of our proposed framework is to detect attacks in transactions and SCs after their deployment in a blockchain network, and before such transactions are validated and added to the main chain. Our proposed framework automatically extracts transaction features in real-time and efficiently analyzes them to detect attacks. To achieve real-time analysis, we collect unvalidated transactions (i.e., pending transactions) in a blockchain system and analyze them to detect attacks before they are validated and added to the main chain. To facilitate this, we first build a highly effective tool, called **Blockchain Code Extraction and Conversion Tool (BCEC)**, to convert important information of pending transactions and SCs to an image form. This tool calls the transaction using its hash (i.e., a feature of the transaction) and then extracts key fields like Bytecode and value from the transactions. After that, it can convert the contents into images for further processing. Second, we develop a CNN-based framework to learn and detect attacks for transactions and SCs. To the best of our knowledge, **this is the first CNN-based framework that analyzes the Bytecode directly and detects various types of attacks in transactions and SCs**. Such a CNN-based framework, which uses important information from transactions for analysis, is more flexible and more effective at detecting new types of attacks than other vector-based methods. To address the third challenge about centralized attack detection, we develop a highly-effective collaborative cyberattack detection framework that can detect cyberattacks inside transactions and SCs in real-time with high accuracy. In our proposed framework, the CNN of each mining node can exchange learning knowledge (i.e., the gradients) with other nodes to create a global model. In this way, the learning model of each node can improve the detection accuracy without sending their local data over the network. Our major contributions can be summarized as follows:

- We implement a blockchain system and perform experiments to build a novel dataset named BTAT. To the best of our knowledge, this is the first dataset with cyberattacks on transactions and SCs of a blockchain system that is synthesized in a laboratory.
- We develop BCEC that can collect transactions, extract their features, and convert them into images to build a dataset. This tool can be implemented in real-time to support the analysis of the attack detection framework.
- We develop a real-time attack detection framework that can be deployed at the mining nodes to detect attacks in transactions and SCs for a blockchain network. In our framework, the mining nodes can detect attacks in transactions and SCs in real-time at about 2,150 transactions per second.
- We propose a collaborative learning framework that can efficiently detect attacks in blockchain networks. In our framework, each mining node can exchange learning knowledge with others and then aggregate a new global model without a centralized model. In this way, our framework can achieve high accuracy in detecting attacks without exposing the mining node's local dataset over the network.
- We perform both simulations and real-time testing to evaluate our proposed framework. Our proposed framework can achieve accuracy up to 94% in simulation and

[1]https://avitech-vnu.github.io/BTAT

91% in real-time experimental results. In addition, our framework has the capacity to analyze various types of transaction features, expanding the detection capabilities for the diversity of attacks.

The rest of this paper is organized as follows. Section II discusses related works. Section III provides the fundamental background of blockchain and our proposed collaborative learning framework. Section IV presents in detail our proposed collaborative framework to detect attacks in SCs and transactions in the blockchain systems. The experiment setup, dataset collection, evaluation methods, simulation and experimental results are described in detail in Section V. We then provide the discussion and our future research directions in Section VI. Finally, we conclude the paper in Section VII.

II. Related work

A. Attacks on Blockchain Network Infrastructure

There are several works trying to deal with attacks on blockchain networks. In [22], the authors propose BrainChain, a machine learning-based method for protecting blockchain systems against distributed denial-of-service (DDoS) attacks. Simulation results show that in a Software-Defined Networking (SDN) environment, BrainChain achieves a detection rate close to 100% with a false positive rate of 21% under a 500 Mbps traffic load. In [23], the authors propose a framework to detect DDoS attacks in a blockchain-based IoT system. Their approach uses a distributed intrusion detection system, deploying Random Forest and XGBoost models at each Fog node to protect IoT sensor data before it is sent to blockchain clouds. Simulation results show that their framework achieves approximately 99% accuracy in detecting attacks. In [24], the authors propose a framework that can deal with multiple attacks in a blockchain network. They used a Deep Belief Network combined with Federated Learning to learn and predict four different types of attacks. Their solution achieved 97.42% accuracy in simulations and 98.61% accuracy in real-time experiments. Although [23], and [24] incorporate decentralized learning models within blockchain environments, and [22] explores the use of Software-Defined Networking (SDN), the primary focus of all three studies is on analyzing network traffic to detect common types of cyberattacks, including DDoS, brute force, and man-in-the-middle attacks. However, these approaches do not specifically address threats that target SCs and transactions, which demand distinct analytical methods and detection mechanisms.

B. Detection of Smart Contract and Transaction Attacks

There are a few works trying to deal with attacks on transactions and SCs in blockchain networks. In [16], the authors propose to convert the source code of SCs into vectors. They then use bidirectional long-short-term memory (LSTM) to identify abnormal patterns of vectors to detect re-entrancy attacks. The simulation results show that their proposed model can achieve 88.26% F1-Score and 88.47% accuracy in detecting re-entrancy attacks. In [17], the authors propose to detect the vulnerabilities inside SCs. To do this, they use feature extraction to analyze the Bytecode of SCs. In [17], the authors use various types of machine learning models to detect 6 types of vulnerabilities with an F1-score of up to 97%. Even though the methods in [16], [17] can detect some types of attacks, they need to use the source code of SCs in high-level programming languages (e.g., Solidity). It is worth noting that when an SC is created, the SC creates corresponding transactions for execution and then sends them to MNs for the mining process. From the MN's point of view, we can only observe transactions with the encoded content (e.g., Bytecode) in their features. In real-time attack detection, we need to analyze this content to find out attacks in transactions and SCs.

C. Bytecode-Level and Symbolic Analysis Methods

Unlike the above deep learning approaches, in [18], the authors propose an intrusion detection system named ContractGuard that can defend Ethereum smart contracts against malicious attacks. The experimental results show that ContractGuard successfully guards against attacks on all real-world vulnerabilities without the need of Solidity source code. However, ContractGuard requires additions of 36.14% gas consumption and 28.27% time consumption compared to those of the original SCs. Moreover, another trade-off of the proposed scheme is that it needs centralized administrations to manually verify malicious transactions. In [19], the authors study the Bytecode. They propose to use the attack vector method to directly analyze the Bytecode. This approach can effectively detect some specific attacks by using pre-defined vectors. However, this method is difficult to extend to various types of new attacks. In addition, even though the attack detection ability can achieve up to 100% in some types of attacks (e.g., re-entrancy, delegatecall, overflow, etc), the authors only test this method on a small scale of data (about 100 samples).

In [14], the authors propose to use Graph embedding to analyze Bytecode. To do this, the authors convert the Bytecode of SC into vectors and then compare the similarities between the vectors of SC to detect SC attacks. The experimental results show that this method can achieve a precision of up to 91.95% in detecting attacks. Both [19] and [14] have to use source code to analyze the bytecode. In [20], the authors introduce an SC security testing approach with the aim of identifying the suspicious behaviors associated with vulnerabilities of SCs in blockchain networks. According to their evaluations, the proposed framework completely rejected about 3.5% of transactions because they could not be tested. Therefore, they point out that further Bytecode analysis can reduce this portion. In addition, in [15], the authors propose DefectChecker to analyze vulnerabilities in SCs. This framework uses symbolic execution to analyze Bytecode without the need for source code. This framework can detect eight types of vulnerabilities in SCs and get an F1-score of 88%.

D. Limitations of Centralized Learning Approaches

All the aforementioned methods focus on centralized learning. They are used to detect attacks in transactions and SCs are summarized in Table I. To implement those methods, all the data needs to be gathered in a centralized server for learning

TABLE I: Summarize recent works on attacks and vulnerabilities detection in transactions and SCs

Work	Year	Method	Dataset	Performance	Remark
P. Qian et al. [16]	2020	Using bidirectional long-short-term memory (LSTM) to analyze	Ethereum mainnet	F1-Score up to 88.26% and accuracy up to 88.47% in detecting re-entrancy attacks	Using source code to analyze
W. Wang et al. [17]	2020	Using source code and various types of machine learning models	Ethereum mainnet	F1-score up to 97%	Using source code to analyze
X. Wang et al. [18]	2019	Proposed ContractGuard that embeds the IDS into EVM binary code. If ContractGuard finds at least one abnormal path, it will raise an alarm to administrators	Deploying using Private Dataset, testing in mainnet	ContractGuard guards against 100% of 6 vulnerabilities in SWC and 83% of the seeded vulnerabilities	Centralized model without source code
Q.-B. Nguyen et al. [19]	2019	Using pre-defined vectors to detect attacks in bytecode	Private Ethereum network	Accuracy up to 100% on 3 vulnerabilities in SWC	Using source code to analyze and test in a small scale with 100 samples
J. Huang et al. [14]	2021	Using Graph embedding to analyze Bytecode	Ethereum mainnet	Precision up to 91.95% over 174 SCs on 5 vulnerabilities in SWC	Using source code to analyze
N. Ivanov et al. [20]	2023	Using the local execution of transactions on a fully-synchronized yet isolated Ethereum node	Ethereum mainnet	TxT prevents 31 out of 37 vulnerabilities (83.8%) in SWC	Rejected about 3.5% of transactions because they could not be tested
J. Chen et al. [15]	2021	DefectChecker which uses symbolic execution to analyze bytecode	Contract Defects [21]	F1 score up to 88% in eight types of attacks	Centralized model without source code
Our work	–	Transforming transactions to images and utilizing a collaborative learning framework to classify attacks	BTAT	Accuracy up to 94% through simulations and 91% in real-time experiments on 6 vulnerabilities in SWC	Decentralized model without source code

and analysis. However, blockchain is a decentralized environment, and MNs are distributed worldwide. Thus, gathering all blockchain data to perform training and testing is impractical. In this paper, we propose a collaborative learning model framework that can detect attacks for transactions and SCs without the need for a centralized server in blockchain networks. In contrast, our work adopts a CNN-based collaborative learning framework that directly analyzes Bytecode without requiring source code.

III. Blockchain System: Fundamental and Proposed Collaborative Learning framework

A. Blockchain

Blockchain technology is a decentralized method for storing and managing data. In a blockchain system, each MN can be used to store and process data. When a Mining Node (MN) receives transactions, it typically groups them into a block as part of the mining process. However, it is worth noting that the consensus mechanism is responsible for managing the rules of the mining process in a blockchain network. There are various types of consensus mechanisms being used in blockchain networks [25]. For example, Ethereum 2.0 uses Proof-of-Stake (PoS) [26] as its consensus mechanism for the mining process. In PoS, a validator, who is responsible for proposing a new block, is randomly selected based on the amount of staked ETH in users' deposits. When the mining process is completed, the valid block is added to the main chain of blocks. After that, the block is irreversible to ensure the integrity of transactions in a blockchain. Another characteristic of blockchain is transparency, which enables all MNs to access the history of transactions within a blockchain network. This transparency ensures that total transaction records are visible to all MNs and promotes trust in the blockchain network. Overall, blockchain possesses numerous valuable characteristics, including decentralization, transparency, immutability, and data tamper resistance, making it applicable across various sectors to enhance human life.

B. Designed Blockchain System and Our Proposed Collaborative Learning Framework

In our laboratory, we set up experiments to collect datasets for training and testing our framework. We first deploy a blockchain system based on a private Ethereum network in our laboratory (more details are shown later in Section V). This experimental setup uses the latest version of the Ethereum network (i.e., Ethereum 2.0). This version uses PoS as a consensus mechanism for validating new blocks. Our system includes various MNs that collect data from their local networks and bootnodes, and the management nodes to connect

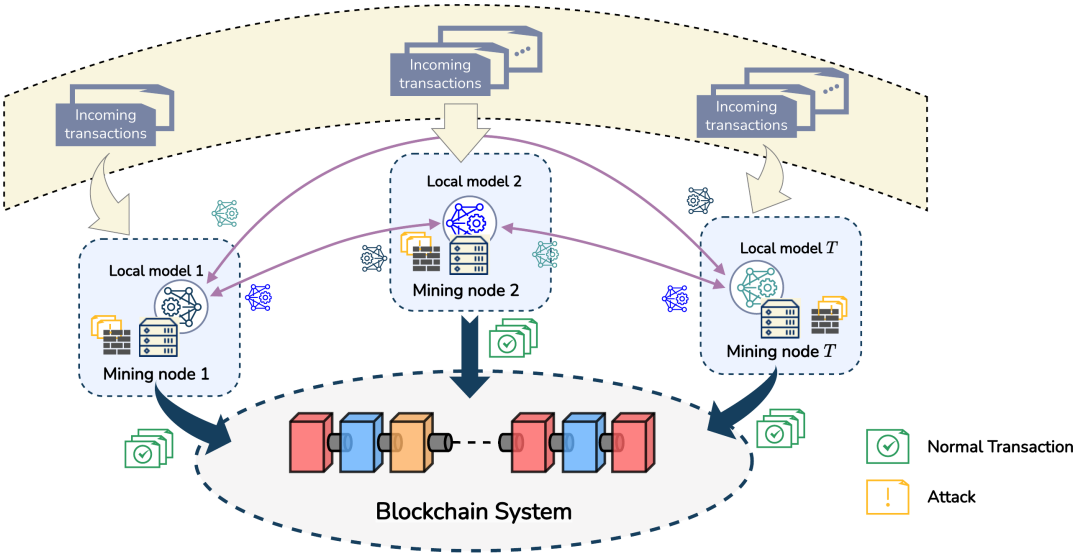


Fig. 1: The system model of our proposed framework. While receiving transactions, our framework will perform preprocessing to extract important information. After that, our collaborative learning will perform the attack detection process to detect network normal behavior or a type of attack.

MNs together. The MNs can receive transactions from various types of blockchain applications, such as smart cities, smart agriculture, IoT, and cryptocurrency. As described above, the transactions are first sent to MNs. They are then put into a block, and the MNs perform the mining process to put them into the main chain. We perform experiments on the system using malicious transactions and SC attacks. These attacks (i.e., DoS with block gas limit, overflows and underflows, flooding of transactions, re-entrancy, delegatecall, and function default visibility) happened and caused serious damage to blockchain systems [27]. Based on these experiments, we construct a comprehensive, state-of-the-art dataset containing both benign and malicious transactions and SC attacks, which is used to evaluate the effectiveness of various attack detection methods.

In this paper, we consider a blockchain system with T MNs working in a blockchain system as described in Fig. 1. When an MN receives pending transactions from the blockchain network, it uses **BCEC** to preprocess them by extracting information from important features and then converting them to grey images. After that, we propose a collaborative learning framework for analyzing the images to detect attacks in transactions and SCs. In our framework, each MN uses its local dataset to train a deep neural network. After the training process, each MN shares its gradient with other nodes and also receives their gradients in return. Afterward, every MN aggregates all the received gradients from other nodes together with its current gradient to generate a new global model for further training (we will explain more details in the next section). In this way, MN can exchange its learning knowledge with the neural network of other MNs. This approach can not only improve the overall learning knowledge of the neural network of all MNs but also protect the privacy of local data over network transmission. By preventing the transmission of the local data of each MN over the network, our approach can also reduce network traffic to avoid network congestion. Thus, the neural networks of MNs can improve the accuracy of detecting attacks for transactions and SCs in blockchain systems.

IV. Proposed Attack Detection Framework

In our proposed attack detection framework, the MNs are used to learn and share their learning knowledge with others to improve the accuracy of their attack detection. At each MN, we propose to use a deep neural network as a detector to learn the data of the MN's local system. After that, the MN exchanges its learning knowledge (i.e., gradient) with other MNs. When an MN receives gradients from others, it will integrate these models with its current model to train its local dataset. This process is iteratively repeated until reaching a predefined number of iterations. Unlike conventional federated learning, which depends on a central server to manage model updates and is at risk of a single point of failure [28], our collaborative learning model is fully decentralized. It allows nodes to share updates directly through peer-to-peer communication, improving fault tolerance and ensuring that training can continue even if some nodes are temporarily offline. In summary, our proposed framework includes three processes as described in Fig. 2. The first process is preprocessing. In this process, our proposed framework captures and extracts the important information of the incoming transactions and then converts them to grey images. The second process is to develop a deep convolutional neural network to classify the grey images to detect attacks. The last process is collaborative learning. In this process, each MN can exchange the gradient with others to improve the accuracy of attack detection.

A. Preprocessing Process

Fig. 2 describes our proposed preprocessing process for transactions in a blockchain system. The main purposes of

This article has been accepted for publication in IEEE Transactions on Cognitive Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCCN.2025.3637274

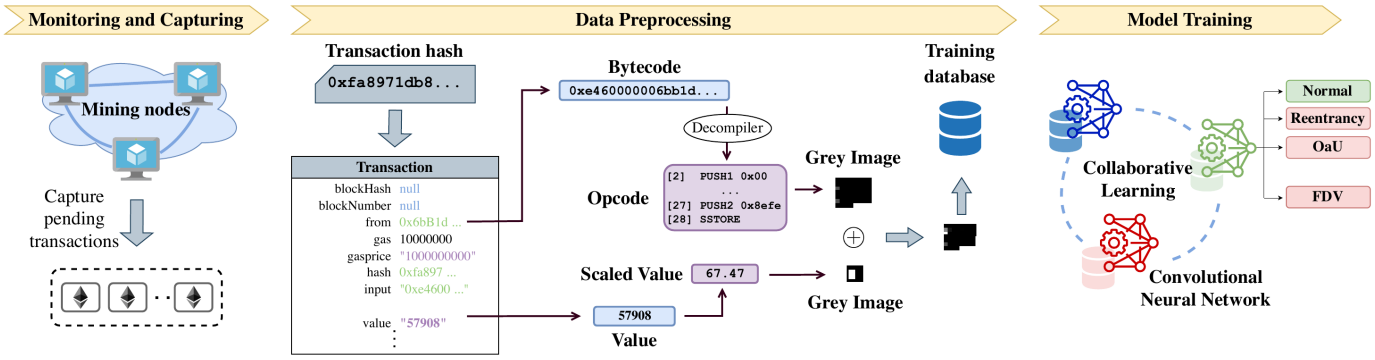


Fig. 2: The processes of our proposed framework: The captured pending transactions are preprocessed using the BCEC tool. This tool extracts important features of transactions and converts them into images. After that, the images are processed by the CNN and collaborative learning models to detect attacks.

the preprocessing process are extracting the important features from incoming transactions and converting them into images for further processing. It is worth noting that SCs are a set of agreements to deploy transactions. For implementation, a server has to send transactions of the SCs to the MN for the mining process. From the perspective of the MN, transactions are visible only as hashes, which are unique identifiers represented as hexadecimal strings. The preprocessing process has three steps to deal with these hashes as follows:

- **Step 1: Capturing data:** Capture hashes from the MN and then recover pending transactions from hashes to have the full information of all transaction features, such as content, value, block hash, block number, chainID, etc.
- **Step 2: Extracting features and converting data:** Extract the content of two crucial features in transactions, named Bytecode and value. The bytecode feature represents the executable functions of a transaction, while the value feature specifies the amount of ETH (Ethereum's native cryptocurrency) transferred. Although we can effectively use the bytecode feature in detecting various types of attacks in transactions and SCs, it does not provide any information on some specific types of attacks, such as Flooding of Transactions [24], where the transaction content is null. Thus, it may be inefficient if we only rely on the bytecode feature for analysis. Therefore, we propose to enhance the attack detection framework by incorporating information from the value feature (its benefits are discussed in more detail in Section V). After that, we apply appropriate preprocessing methods to the corresponding features as follows:
  - **Bytecode feature**: Extract the content and then transform it into opcode using EVM Bytecode Decompiler [29]. The opcode is a series of executed comments in assembly. Thus, we propose to convert all features of this assembly code to a grey image named Grey Image 1.
  - **Value feature**: we first scale its content to an appropriate range and then convert it to another grey image named Grey Image 2.
- **Step 3: Combining data:** In this step, we combine both Grey Image 1 and Grey Image 2 to create the Final Grey Image. This Final Grey Image includes all essential information of a transaction and an SC in the blockchain system. They can be used to train the deep convolutional neural network to detect attacks inside.

In this framework, all these steps are encapsulated in the **BCEC** tool. This tool performs the preprocessing process in real-time to support the analysis of collaborative attack detection for transactions and SCs in a blockchain system.

B. Learning Process

1) CNN Architecture for Attack Detection: In our proposed framework, at each MN, we implement a detector that can help to detect attacks based on the transformed images from the preprocessing process with high accuracy. The core component of the detector is developed based on a Deep CNN. The reason for using CNN is that this framework can classify a large amount of labeled data, especially in image classification with high accuracy [30]. Additionally, in our proposed approach, the CNN model does not have to learn their local data separately; it can exchange its gradient with other MNs to improve the learning knowledge as well as enhance the accuracy of attack detection. In detail, the architecture of CNN in an MN includes three types of layers, i.e., convolutional layer, max pooling layer, and fully connected layer [30]. These layers are described as follows:

- **Convolutional layer:** The neurons in this layer are formed in feature maps to learn the feature representation of the input. In addition, these feature maps can connect with others of the previous layer by weight parameters called filter banks [31]. In this layer, the input data is convolved with weight parameters in every iteration to create feature maps.
- **Max pooling layer:** The main purpose of this layer is to reduce the resolution of feature maps in the previous layer. To do this, this layer selects the largest values in areas of the feature map [30] and then sends them to the next layer.
- **Fully connected layer:** This layer performs classification functions for the neural network. In this layer, the feature maps from previous layers are first flattened. They are then put into a fully connected layer for classification. The softmax function is included at the end of this layer to produce the output prediction.

2) Local Model Training Process: We denote $\boldsymbol{D}$ as a local dataset of an MN to train a CNN. $\boldsymbol{D}$ includes $\boldsymbol{S}$ images and $\boldsymbol{Y}$ labels so we can denote $\boldsymbol{D} = (\boldsymbol{S}, \boldsymbol{Y})$. We consider $n = \{1, .., N\}$ as the training layer of the neural network. We denote N as the number of training layers of the neural network. We denote $\boldsymbol{I}$ as the matrix features of image $\boldsymbol{S}$, and $\boldsymbol{I_i}$ as the matrix features of image $\boldsymbol{S}$ at iteration i. We denote T as the total number of MNs and $t \in T$ as the MN number. The output of a convolutional layer n, $n \in \{1, .., N\}$, at iteration i can be calculated as follows [32]:

$$\boldsymbol{I}_{n+1,t,i} = \gamma_{n,t}\Big(\boldsymbol{I}_{n,t,i} * \boldsymbol{F}_{n,t}\Big), \tag{1}$$

where $(*)$ is the convolutional operation, $\gamma_{n,t}$ is the activation function and $\boldsymbol{F}_{n,t}$ is the filter bank of layer n of MN t. After that, the output of the convolutional layer is put into a max pooling layer. The output of a max pooling layer can be calculated as follows:

$$\boldsymbol{I}_{n+2,t,i} = \varphi_t\Big(\boldsymbol{I}_{n+1,t,i}\Big), \tag{2}$$

where φ_t is the max pooling function that selects the maximum value in a pooling area. We denote $\boldsymbol{I}_{e,t,i}$ as the matrix features of the last image after processing with multiple convolutional layers and max pooling layers. $\boldsymbol{I}_{e,t,i}$ is put into a softmax function to classify and produce the output in the fully connected layer. We consider $l \in \{1, ..., L\}$ as the classification group number, $\hat{Y}_l \in \hat{\boldsymbol{Y}}$ as the output prediction, the probability that an output prediction $\hat{Y}$ belongs to group l can be calculated as follows:

$$\begin{aligned} p(\hat{Y}_l = l|\boldsymbol{I}_{e,t,i}, \boldsymbol{W}_{e,t,i}, \boldsymbol{b}_{e,t,i}) \\ &= \text{softmax}(\boldsymbol{W}_{e,t,i}, \boldsymbol{b}_{e,t,i}) \\ &= \frac{\exp\left(\boldsymbol{W}_{e,t,i}\boldsymbol{I}_{e,t,i} + \boldsymbol{b}_{e,t,i}\right)}{\sum_l \exp\left(\boldsymbol{W}_{e,l,t,i}\boldsymbol{I}_{e,t,i} + \boldsymbol{b}_{e,l,t,i}\right)}, \end{aligned} \tag{3}$$

where $\boldsymbol{W}_{e,t,i}, \boldsymbol{b}_{e,t,i}$ are the weights and biases of the fully connected layer at iteration i of MN t, respectively; and $\boldsymbol{W}_{e,l,t,i}, \boldsymbol{b}_{e,l,t,i}$ as weights and biases of the fully connected layer at iteration i to classify an output prediction into class l. In other words, equation (3) computes the probability that a given input image belongs to each possible attack class by weighing its learned features and normalizing the results so that the highest value indicates the most likely classification. Based on equation (3), we can calculate a vector of prediction $\hat{\boldsymbol{Y}}$ which includes output prediction $\hat{Y}_l$ belonging group l with probability p as follows:

$$\hat{\boldsymbol{Y}} = \underset{l}{\text{argmax}}[p(\hat{Y}_l = l|\boldsymbol{I}_{e,t,i}, \boldsymbol{W}_{e,t,i}, \boldsymbol{b}_{e,t,i})]. \tag{4}$$

In this stage, we compare the output predictions with the labels using a sparse categorical cross-entropy function to calculate the loss for backpropagation. We denote $Y_l \in \boldsymbol{Y}$ as the label of class l in $\boldsymbol{Y}$. The loss function can be calculated as follows:

$$\boldsymbol{J}(\boldsymbol{W}) = -\sum_{l=1}^{L} Y_l \log \hat{Y}_l. \tag{5}$$

This equation measures how far the predicted outputs deviate from the true class labels. A higher loss indicates a larger error, while a lower loss means the model's predictions are closer to the actual labels. We denote $\boldsymbol{W}$ as the model of the neural network. Based on equation (5), we can calculate the gradient of this function as follows:

$$\nabla\boldsymbol{\theta}_{t,i} = \frac{\partial \boldsymbol{J}(\boldsymbol{W})}{\partial \boldsymbol{W}} = -\frac{\partial\Big(\sum_{l=1}^{L} Y_l \log \hat{Y}_l\Big)}{\partial \boldsymbol{W}}. \tag{6}$$

After obtaining the gradient based on equation (6). We then use it for the Adam optimizer to update the parameters of the neural networks. We consider $m_{t,i+1}$ and $v_{t,i+1}$ as the moment vectors of the next iteration $i+1$ of the Adam optimizer. The $m_{t,i+1}$ and $v_{t,i+1}$ can be calculated from the gradient and Adam functions [33] as $m_{t,i+1} = A_1(\nabla\boldsymbol{\theta})$ and $v_{t,i+1} = A_2(\nabla\boldsymbol{\theta})$. We denote $\boldsymbol{\Gamma}_{t,i}$ as a trained model, and $\theta_{t,i}$ as a global model at iteration i. With $\beta_{t,i+1}$ as the learning rate, a newly trained model at the next iteration $i+1$ can be calculated as follows:

$$\begin{aligned} \boldsymbol{\Gamma}_{t,i+1} &= \boldsymbol{\Gamma}_{t,i} - \beta_{t,i+1}\frac{m_{t,i+1}}{\sqrt{v_{t,i+1}}} \\ &= \boldsymbol{\Gamma}_{t,i} - \beta_{t,i+1}\frac{A_1(\nabla\boldsymbol{\theta}_{t,i})}{\sqrt{A_2(\nabla\boldsymbol{\theta}_{t,i})}}. \end{aligned} \tag{7}$$

C. Collaborative Learning Process

In this paper, we propose a Collaborative Deep Convolutional Neural Network framework (Co-CNN) to detect the different types of attacks in a blockchain network. In this framework, each MN has a CNN model to train and test its dataset. The CNN model can receive gradients from other MNs to improve the accuracy of attack detection. To do this, the CNN model of an MN first gets the gradient based on equation (6). It then sends the gradient to other MNs and receives gradients from others. We consider at iteration i, an MN receives $T-1$ gradients from others. $\nabla\boldsymbol{\theta}_{t,i}$ is the gradient of MN t at iteration i. It can aggregate all gradients using the following formula [34]:

$$\nabla\boldsymbol{\theta}_{t,i+1} = \frac{1}{T}\sum_{t=1}^{T} \nabla\boldsymbol{\theta}_{t,i}, \tag{8}$$

where $\nabla\boldsymbol{\theta}_{t,i+1}$ is the new aggregated gradient. After generating a new aggregated gradient, each MN will calculate a newly trained model using equation (7). This process continuously repeats until the algorithm converges or reaches the predefined maximum number of iterations. After the training process, we can obtain the optimal trained model in each MN to analyze and detect attacks inside a series of grey images. This process is summarized in Algorithm 1.

V. Experiment and Performance Analysis

A. Experiment Setup

In our experiments, we set up an Ethereum 2.0 system in our laboratory, as shown in Fig. 3. This version of Ethereum uses a new consensus mechanism, namely PoS instead of Proof-of-Work (PoW). In our experiments, there are five Ethereum nodes, two bootnodes, a trustful device, and an attack device. All these devices are connected to a Cisco switch, which acts

Algorithm 1 The learning process of the Co-CNN model

while $i \leq$ maximum number of iterations **do**
  for $\forall t \in T$ **do**
    The CNN of an MN t learns $\boldsymbol{D}_t$ to generate output $\hat{Y}$.
    The MN t generates gradient $\nabla \boldsymbol{\theta}_{t,i}$ and sends it to others.
    The MN t receives $T-1$ gradients from others.
    MN calculates a new optimal trained model $\boldsymbol{\Gamma}_{t,i+1}$.
  end for
  $i = i + 1$.
end while
MN uses its optimal model $\boldsymbol{\Gamma}_{optimal}$ to detect attacks based on the input images.

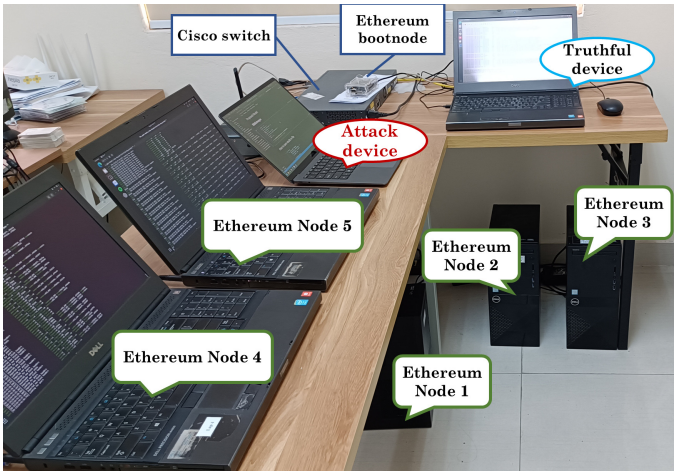


Fig. 3: Real-time experiment setup.

as the central hub for our local network. The configuration of these devices is as follows:

- Ethereum nodes are launched using *Geth v1.10.22*, an official open-source implementation of Ethereum network [35], and *Prysm v3.2.0*, an official implementation of the PoS consensus mechanism in Ethereum 2.0 [36]. They share the same genesis configurations, e.g., chainID, block gas limit at 30,000,000 gas, etc. Nodes 1, 2, and 3 are configured on workstation computers with Intel Core i9-10900 processors at @5.2 GHz and 64 GB of RAM. Nodes 4 and 5 are configured on personal computers with Intel Core i7-4810MQ processors at @3.8 GHz and 16 GB of RAM.
- *Geth* bootnode and *Prysm* bootnode are also created by *Geth v1.10.22* and *Prysm v3.2.0*, respectively. They are responsible for connecting all the Ethereum nodes together.

B. Dataset Collection

According to the detailed analysis of the public Ethereum network on transaction behavior [37], the addresses that are associated with fewer than 10 transactions account for 88% of total addresses. About 50% received addresses appear only one time for a transaction in history. This is because most people want to create transactions anonymously. Therefore, to increase the diversity and realism of our dataset, we generate a pool of 10,000 unique Ethereum accounts, which are used to simulate transactions directed to various nodes in the network. In order to construct the normal state in the BTAT dataset, a truthful server, as shown in Fig. 3, randomly selects accounts from these accounts to create transactions for the blockchain system. We deploy an attack device within a controlled laboratory Ethereum environment to construct the attack states in the dataset. This device is used to perform multiple types of attacks to simulate realistic malicious behavior. The resulting transactions are captured using our data collection tool, which automatically collects and labels both benign and malicious activities. In practice, each real-world attack is often represented by only one or a few recorded transactions. To improve the quality and representativeness of the dataset, we extend our data collection tool to automatically generate multiple samples from the same type of attack by interacting with different SCs and executing various transaction functions. The BTAT dataset includes seven states: one normal state and six attack states, described as follows:

1) Normal State: For the normal state, we use *OpenZeppelin Contracts* [38] library as the secured SCs. Two types of transactions below are used to generate samples randomly for the normal state.

- Exchange ETH: On the public Ethereum network, most transactions only exchange ETH to another address without any bytecode. This kind of transaction accounts for 75% of the total samples of the normal state in our experiment.
- Transactions-related SCs: There are two types of these transactions. The transactions for deploying SCs and the transactions that interact with functions in deployed SCs. We perform three essential SCs' categories in the Ethereum system, i.e., Tokens/Coins/NFT, Ethereum 2.0 deposit, and SCs for other purposes.

Although the number of original SCs is small compared to the total transactions in the dataset. The contents of transactions and deployed SCs are not duplicated. The reason is that we randomly select not only the senders and recipients but also the amount of ETH and inputs of functions in any generated transaction.

2) Attack States: SCs have a number of vulnerabilities listed in SWC [7] because of programmers, consensus mechanisms, and compilers. Attackers can exploit these weaknesses of SC to perform attacks and then steal money in blockchain systems [27]. In BTAT, we regenerate several real-world attacks from the tracks that they left on Ethereum's ledger. We give a brief description of the six types of attacks.

- *DoS with Block Gas Limit (DoS)*: There are several functions inside SCs. These functions can be temporarily disabled when their gas requirements exceed the block gas limit. A notable *DoS* incident occurred in 2015 when the payout function of the GovernMental SC, which held a jackpot of 1,100 ETH, became stuck due to this issue [7]. In our work, we deploy the GovernMental SC and repeatedly join the jackpot to intentionally trigger this vulnerability and disable the payout function."

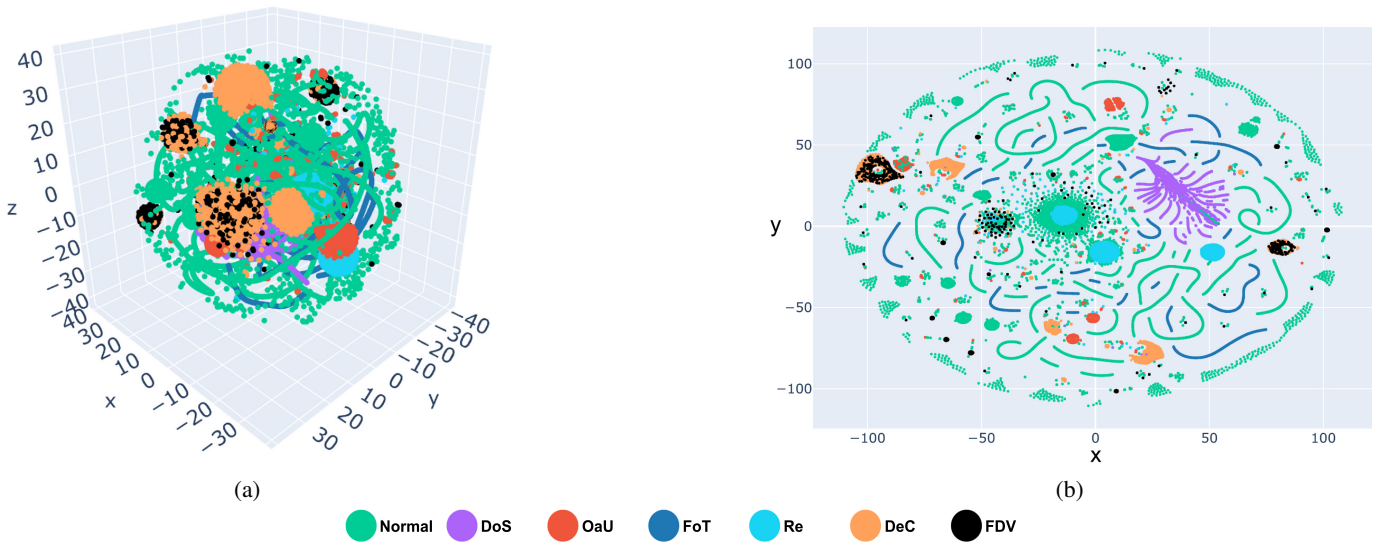


Fig. 4: Visualization using t-SNE of BTAT dataset: (a) Visualization in 3D. (b) Visualization in 2D.

TABLE II: Number of samples on the proposed BTAT dataset.

Class	Number of samples	Portion (%)
Normal	152,423	50.34
DoS	22,994	7.59
OaU	29,254	9.66
FoT	41,732	13.78
Re	22,682	7.49
DeC	22,455	7.41
FDV	11,209	3.73
Total	302,749	100

- *Overflows and Underflows (OaU)*: In solidity language, if a variable is out of its range, it is in the overflow or underflow state. In this case, the variable is turned to another value (e.g., 0 for overflow and $2^{256}-1$ for underflow). Attackers can use this vulnerability to bypass SCs' conditions when withdrawing funds. For example, they can bypass the requirements of checking their accounts' balances. Several real *OaU* attacks were detected, e.g., 2^{256} BEC tokens, CSTR token, USD $800k of PoWH token [11], and so on [7]. We replicate the above *OaU* attacks on their original SCs in the dataset.
- *Flooding of Transactions (FoT)*: Attackers spam a number of meaningless transactions to delay the consensus of blockchain networks. Such an attack caused the unconfirmation of 115k Bitcoin transactions in 2017 [24]. In our setup, *FoT* attacks are generated by continuously sending a negligible amount of ETH from a random sender to another arbitrary recipient.
- *Re-entrancy (Re)*: When the SCs do not update their states before sending funds, attackers can recursively call the withdraw function to drain the SCs' balances. Two types of *Re* are single-function and cross-function. The single-function type happened and led to a loss of 3.6 million ETH in 2016. Both types of Re are performed in our dataset [7].
- *Delegatecall (DeC)*: *delegatecall()* is the mechanism to inherit functions, storage, and variables from other deployed SCs. If the inherited SCs are attacked, they will indirectly affect the main SC. To implement, we re-create the 2nd Parity MultiSig Wallet attack [7]. In this attack, attackers took control and suicided the inherited SCs.
- *Function Default Visibility (FDV)*: If the programmers do not define the visibility of functions in SCs, it will default to public. Thus, anyone can interact with those functions. For implementation, we perform the 1st Parity MultiSig Wallet attack [7]. In this attack, attackers took control of this SC through an *FDV* flaw.

Table II shows the number of samples in each class of our proposed BTAT dataset. The proportions of the samples in the classes are not balanced, e.g., the number of *Re* samples is twice that of *FDV*. Because *Re* requires a series of malicious transactions instead of only one malicious transaction as in *FDV*. We use *t*-Distributed Stochastic Neighbor Embedding (*t*-SNE) [39] to visualize our designed BTAT in 3D and 2D, as shown in Fig. 4. In this figure, the points of all classes are randomly scattered and form non-linear lines. Additionally, we can see in Fig. 4(b) that numerous points of *FDV* and *DeC* overlap with each other, and the same scenario occurs with *Normal* and *Re*. These overlaps create significant challenges for attack detection in the next section.

C. Evaluation Methods

The confusion matrix [40], [41] is widely used to evaluate the performance of machine learning models. We denote TP, TN, FP, and FN as "True Positive", "True Negative", "False Positive", and "False Negative". In this paper, we use ubiquitous parameters (i.e., accuracy, precision, and recall) in the confusion matrix to evaluate the performance of models. The accuracy of a model can be calculated as follows:

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}}. \tag{9}$$

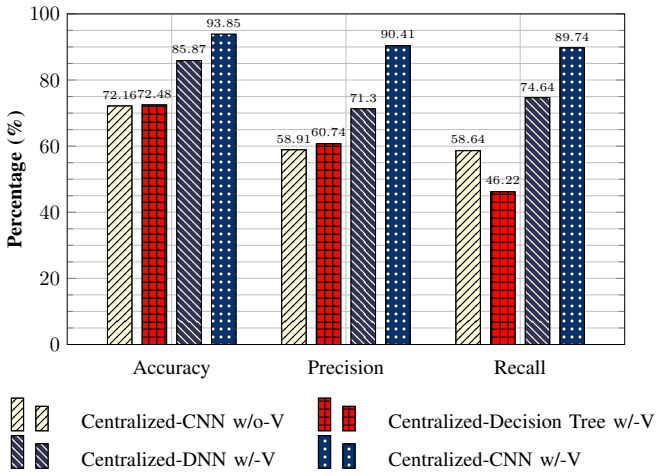


Fig. 5: The results of the preprocessing processes in different schemes.

In addition, we use the macro-average precision and macro-average recall to evaluate the performance of the models. With L as the number of classification groups (i.e., the total number of normal and attack states), the macro-average precision is calculated as follows:

$$\text{Precision} = \sum_{l=1}^{L} \frac{\text{TP}_l}{\text{TP}_l + \text{FP}_l}. \tag{10}$$

The macro-average recall of the total system can be calculated as follows:

$$\text{Recall} = \sum_{l=1}^{L} \frac{\text{TP}_l}{\text{TP}_l + \text{FN}_l}. \tag{11}$$

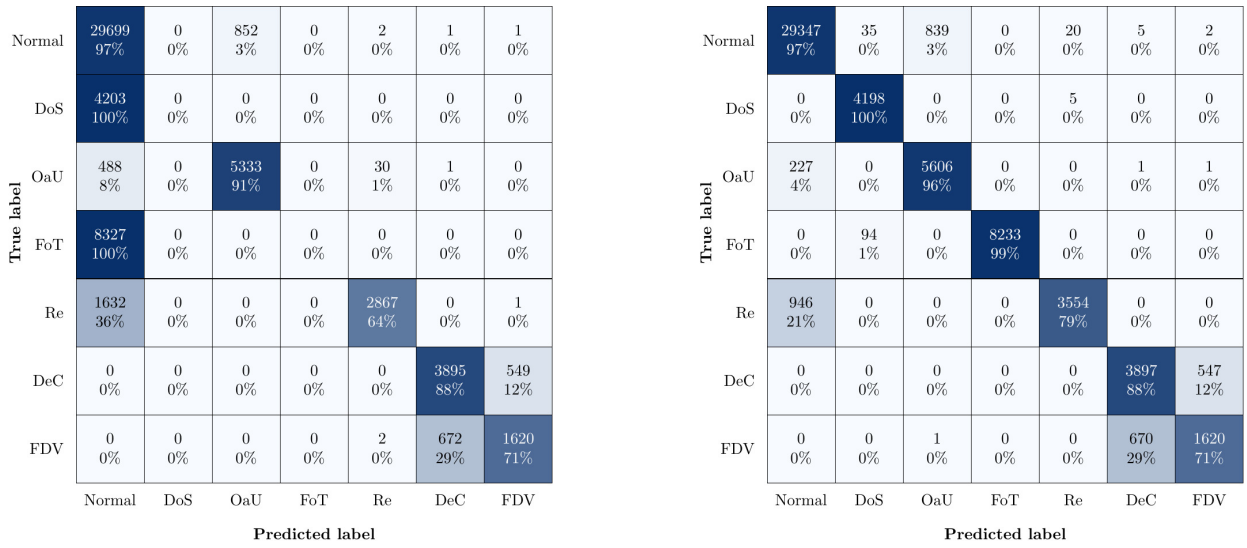
D. Simulation and Experimental Results

In this section, we present the simulation and real-time experimental results of our experiments. In particular, we use the confusion matrix to evaluate our proposed model's performance (in terms of accuracy, precision, and recall) compared to the centralized model.

1) Preprocessing Analysis: In this section, we compare our proposed model in various schemes. On the one hand, we use our proposed preprocessing process as in Fig. 2 under different schemes such as CNN, Deep Neural Network (DNN), and Decision Tree (DT) to compare their evaluation results. On the other hand, we eliminate the value feature and use only the Bytecode preprocessing and the CNN to analyze the transactions and SCs. Through the results of these schemes, we demonstrate the efficiency of our proposed preprocessing process in combining various features of transactions. Fig. 5 describes the evaluation results of these schemes. In this figure, the model w/-V has accuracy, precision, and recall at 93.849%, 90.413%, and 89.742%, respectively. These results outperformed the model w/o-V, which has accuracy, precision, and recall at 72.163%, 58.911%, and 58.638%, respectively. The DNN and DT schemes with Value features achieve accuracies of 85.87% and 72.48%, respectively. They are higher than that of the CNN without Value feature at 72.16%, but lower than that of the CNN with Value feature at 93.85%. Especially, Fig. 6 provides detailed information for all types of attacks and the normal behavior of MN-5. In Fig. 6 and Fig. 7, the diagonal numbers represent the count and percentage of correctly identified samples for each category, while the off-diagonal numbers indicate misdetections. In Fig. 6(a), we can observe that the model w/o-V cannot detect DoS and FoT attacks because it classifies all samples of DoS (row 2, column 1) and FoT attacks (row 4, column 1) into normal behavior. In contrast, the model w/-V in Fig. 6(b) can detect these types of attacks with high accuracy at about 100% for DoS detection (row 2, column 2) and 99% for FoT detection (row 4, column 4). This is because the value feature is essential to support the learning models in detecting many types of important attacks.

2) Accuracy Analysis: In this section, we perform experiments to compare the performance results of the centralized model with our proposed model. The centralized model (Centralized-CNN) that we design can learn knowledge from all MNs for training and testing processes. Besides, we use different schemes of the collaborative learning model with 3 mining nodes (Co-CNN-3), 5 mining nodes (Co-CNN-5), and 10 mining nodes (Co-CNN-10). In each scheme, the collected datasets are divided equally among all mining nodes. To implement experiments, we first perform cyberattacks on transactions and SCs in our deployed private Ethereum platform to collect datasets from all MNs. In our proposed collaborative learning model, each MN uses its local dataset for both training and testing processes. However, in the training process, the MNs can exchange their gradients with others to improve their learning knowledge as well as the accuracy of attack detection. On the other hand, in the Centralized-CNN, all the local datasets of MNs will be gathered into a big dataset for its training and testing process.

The performance results of two scenarios of preprocessing processes (i.e., without value feature (w/o-V) and with value feature (w/-V) with all schemes are also provided in Table III and Table IV. Table III presents the performance of the simulation results of all schemes with the w/o-V preprocessing process. In Table III, the accuracy, precision, and recall are nearly the same at around 72-73%, 58-59%, and 58-59%, respectively. In contrast, in Table IV, we can observe that the performance of all schemes with the w/-V preprocessing process outperforms those w/o-V preprocessing process at about 93-94%, 90-91%, and 89-90% in accuracy, precision, and recall, respectively. In detail, we first can see in Table IV that the performance results of our proposed models are nearly the same as the Centralized-CNN. However, in some MNs, such as MN-5 of the Co-CNN-5, the accuracy, precision, and recall are higher than those of the Centralized-CNN at around 0.6%, 0.6%, and 0.7%, respectively. Specifically, Fig. 7 provides detailed information for each type of attack of Co-CNN-5 in the MN-5. These figures show that the misdetection of the Co-CNN-5 is significantly reduced compared to the Centralized-CNN (as in Fig. 6(b)). In detail, the misclassification rate of the MN-5 from FDV to DeC is at about 27%, which is smaller than that of the Centralized-CNN at 29%. Similarly, the misdetection of the MN-5 from OaU to Normal is at 1% of total samples of OAU, which is smaller than that of the



(a) (b)

Fig. 6: The detection results of the models w/ and w/o-V feature. (a) Centralized-CNN w/o-V. (b) Centralized-CNN w/-V.

TABLE III: Simulation results w/o-V with Centralized-CNN, Co-CNN-3, Co-CNN-5, and Co-CNN-10 models.

	Centralized-CNN	Co-CNN-3			Co-CNN-5				
		MN-1	MN-2	MN-3	MN-1	MN-2	MN-3	MN-4	MN-5
Accuracy	72.163	71.686	71.761	72.080	72.735	72.519	72.211	72.760	72.627
Precision	58.911	58.323	58.298	58.646	59.676	59.300	58.818	59.699	59.032
Recall	58.638	57.539	57.951	58.608	58.955	58.807	58.415	59.444	58.969

	Co-CNN-10									
	MN-1	MN-2	MN-3	MN-4	MN-5	MN-6	MN-7	MN-8	MN-9	MN-10
Accuracy	72.768	73.333	73.184	73.117	73.150	72.984	73.017	73.267	73.516	73.117
Precision	58.169	59.462	59.107	58.957	58.779	58.621	58.288	59.503	59.013	59.125
Recall	58.131	58.531	58.462	58.727	58.775	58.285	58.528	59.066	59.192	58.650

TABLE IV: Simulation results w/-V with Centralized-CNN, Co-CNN-3, Co-CNN-5, and Co-CNN-10 models.

	Centralized-CNN	Co-CNN-3			Co-CNN-5				
		MN-1	MN-2	MN-3	MN-1	MN-2	MN-3	MN-4	MN-5
Accuracy	93.849	93.88	94.384	94.115	94.347	94.057	94.148	94.206	94.439
Precision	90.413	90.216	91.162	90.860	90.794	90.540	90.637	90.903	91.029
Recall	89.742	89.665	90.688	89.970	90.329	89.932	90.025	90.514	90.536

	Co-CNN-10									
	MN-1	MN-2	MN-3	MN-4	MN-5	MN-6	MN-7	MN-8	MN-9	MN-10
Accuracy	93.633	94.248	93.849	93.566	93.899	93.832	93.516	93.732	93.699	93.849
Precision	89.326	90.611	90.095	89.969	90.106	90.048	89.252	90.684	89.778	90.464
Recall	89.206	89.716	89.313	89.114	89.745	89.289	89.213	89.464	89.298	89.477

Centralized-CNN at 4%.

3) Convergence Analysis: In this section, we compare the convergence of different models, i.e., the Centralized-CNN, and the collaborative model with 3, 5, and 10 mining nodes. Fig. 8 describes the accuracy and loss of these models in 1,000 iterations. In general, all of the models converged after about 800 iterations in terms of accuracy and loss. While the accuracies of Centralized-CNN, Co-CNN-3, and Co-CNN-5 models quickly reach the convergence after 400 iterations at about 93%, the accuracies of Co-CNN-10 need about 800 iterations to converge and reach 93%. The same trends happen with the loss. This is because the number of samples of each MN in Co-CNN-10 is much smaller than that of other models, while the number of workers is higher than that of other models. Thus, Co-CNN-10 needs more time to exchange learning knowledge with other models. It finally reaches convergence after about 800 iterations and has an accuracy nearly the same as other models.

4) Real-time Attack Detection: In this section, we consider a practical scenario by evaluating the performance of the system in real-time cyberattack scenarios. To do this, we first take the trained models from all schemes (noted that the trained models are trained in the schemes as in the accuracy analysis, i.e., Centralized-CNN, Co-CNN-3, Co-CNN-5). There are 5 blockchain nodes participating in these experiments, and they join a private Ethereum network as described in the above section. After the learning models are trained, they are deployed on MNs. In the experiments, both

TABLE V: Real-time experiment results.

(a) Centralized-CNN and Co-CNN w/-V

	Centralized-CNN					Co-CNN-3					Co-CNN-5				
	MN-1	**MN-2**	**MN-3**	**MN-4**	**MN-5**	**MN-1**	**MN-2**	**MN-3**	**MN-4**	**MN-5**	**MN-1**	**MN-2**	**MN-3**	**MN-4**	**MN-5**
Accuracy	89.603	89.542	89.668	89.702	89.291	88.663	88.582	88.655	88.794	88.471	90.928	90.896	90.957	91.061	90.614
Precision	76.851	75.806	76.956	76.690	75.582	76.755	75.872	76.845	77.191	75.912	80.192	78.835	80.469	80.846	78.576
Recall	76.858	77.117	76.888	76.767	76.822	78.523	78.939	78.531	78.563	78.724	78.870	79.044	78.757	78.762	78.747

(b) Centralized-CNN and Co-CNN w/o-V

	Centralized-CNN					Co-CNN-3					Co-CNN-5				
	MN-1	**MN-2**	**MN-3**	**MN-4**	**MN-5**	**MN-1**	**MN-2**	**MN-3**	**MN-4**	**MN-5**	**MN-1**	**MN-2**	**MN-3**	**MN-4**	**MN-5**
Accuracy	65.877	65.780	65.643	66.312	65.734	66.804	66.640	66.569	67.115	66.797	65.606	65.579	65.512	66.212	65.830
Precision	47.263	46.105	46.739	47.544	46.012	51.442	50.024	51.116	51.641	50.433	44.668	44.078	44.534	44.994	44.141
Recall	51.383	51.576	51.434	51.318	51.427	49.670	49.888	49.586	49.557	49.625	48.447	48.544	48.381	48.372	48.518

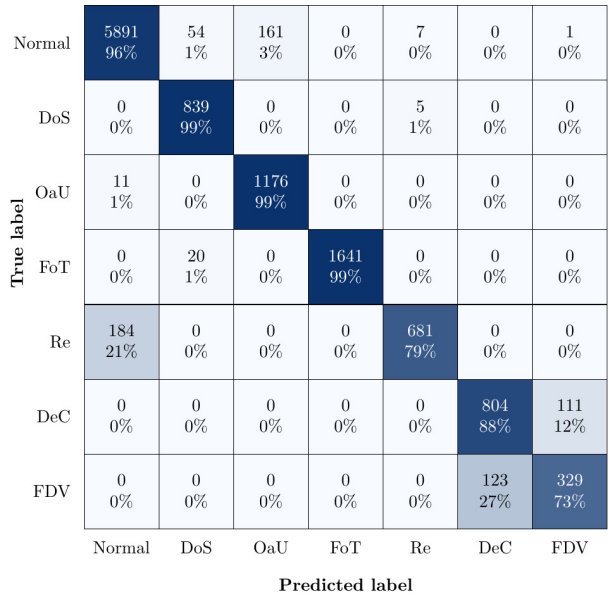


Fig. 7: The detection results of the Co-CNN-5 w/-V.

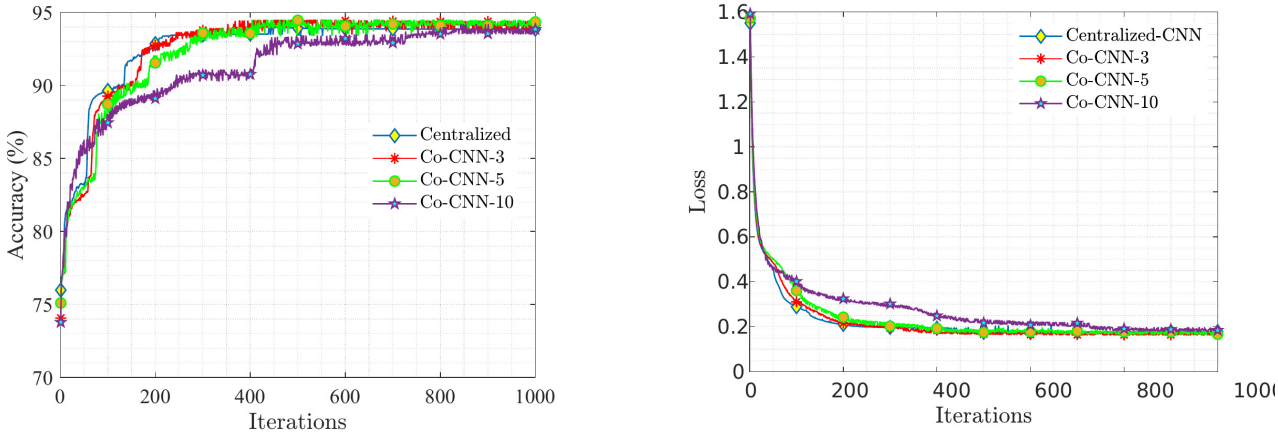
cases with value and without value preprocessing processes are considered. In real-time scenarios, both normal and attack samples continuously come to the blockchain node. To handle this incoming data efficiently, the BCEC employs a pipelined processing technique. It collects all pending transactions in continuous 3-second intervals, with each interval forming a separate batch for analysis. This process runs without interruption, meaning a new batch begins immediately after the previous one ends. At the same time, once a batch is collected, a parallel task is triggered to process it. This task includes preprocessing (converting the transaction data into image representations) and running model predictions. To ensure real-time performance, all processing for each batch must be completed within the same 3-second window, before the next batch is ready.

Table V presents the performance of Co-CNN-3, Co-CNN-5, and Centralized-CNN models in two cases of preprocessing. In general, we can observe in Table V(a) that the performance of these models in accuracy, precision, and recall w/-V in the preprocessing process is at about 88-91%, 76-80%, and 77-79%, respectively. These results outperform those of the w/o-V in the preprocessing process with accuracy, precision, and recall at about 65-66%, 44-51%, and 48-51%, respectively. In addition, when we compare the same case w/-V in the preprocessing process of the simulation as in Table IV and the real-time experimental results as in Table V(a), we can observe that the accuracy, precision, recall of the real-time experimental results are little smaller than those of simulation results about 3%, 10%, and 11%, respectively. This is because, in simulation, we implement multiple types of attacks on the blockchain system and then collect data to have enough samples for the dataset to train the model. However, in real-time scenarios, some attack types, such as Re, DeC, and FDV, rarely appear during the experiment. Thus, it makes it more difficult for the learning models to detect them in real-time. Specifically, we can observe in Table V(a) that MN-4 of Co-CNN-5 has higher performance in accuracy, precision, and recall than MN-4 of the Centralized-CNN by about 1.3%, 4%, and 2%, respectively. Therefore, in real-time detection scenarios, our proposed model still demonstrates better performance in detecting attacks than in simulation.

5) Real-time Monitoring and Detection: Fig. 9 shows the real-time cyberattack monitoring from the output of our proposed model Co-CNN-5 in Ethereum node 1. In these figures, the normal and each type of attack are displayed in different lines. Fig. 9(a) displays the normal state of the system with the high value of the predicted normal state over time. We can observe that in the normal state, the predicted states of all types of attacks are nearly 0. When a type of attack happens, the predicted state of that attack will increase, e.g., the FoT attack state as in Fig. 9(d). As described in the previous section, in real-time scenarios, Re, Dec, and FDV attack states have a small number of attack samples. Therefore, their predicted states in Fig. 9(b), Fig. 9(f) and Fig. 9(g) do not have high values. However, our proposed model can still detect all of the attacks in real-time with high accuracy at 91%.

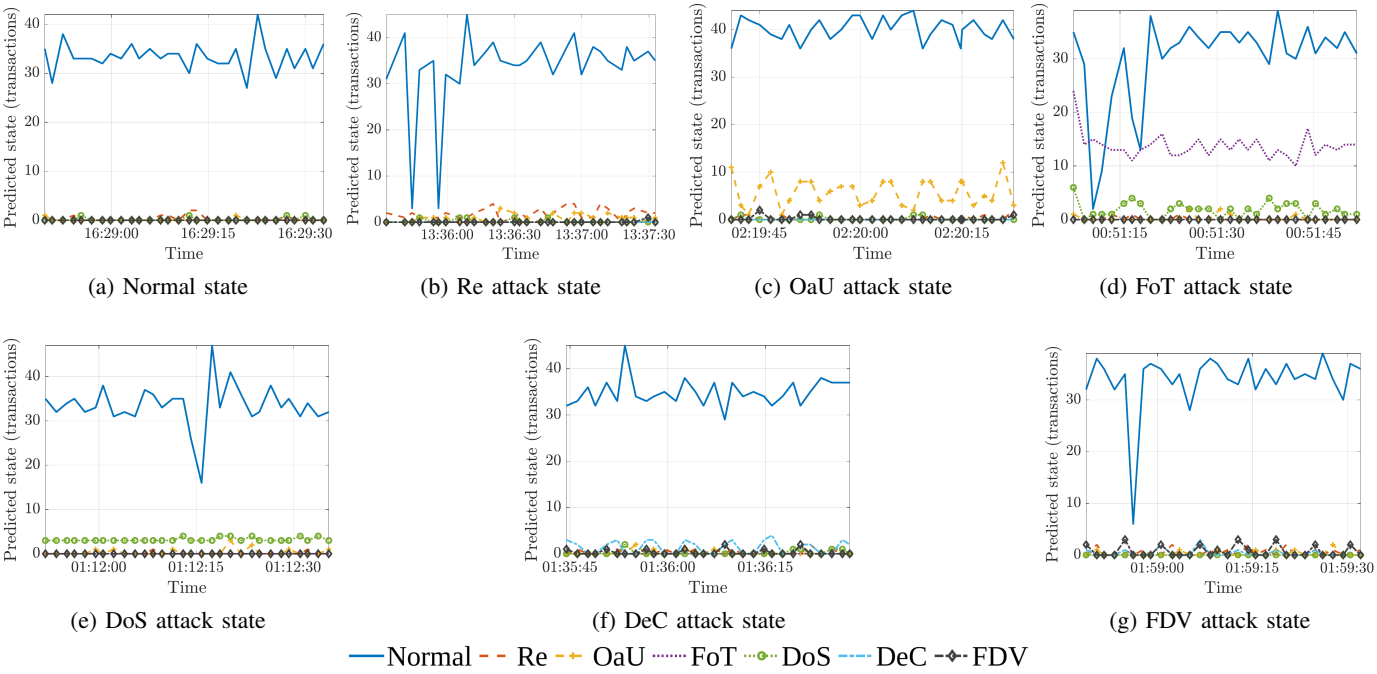
6) Throughput Analysis of Mining Nodes: Fig. 10 describes the processing time of two MNs with the same Co-CNN-5 model. We can observe in Fig. 10 that when the number of transactions increases, the processing time of both MNs also linearly increases. However, there is a difference in capacity between the two MNs. In detail, while MN-5 can process about 1,100 transactions per second, the number of transactions that MN-1 can process is around 2,150 transactions per second.

This article has been accepted for publication in IEEE Transactions on Cognitive Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCCN.2025.3637274



(a) (b)

Fig. 8: The convergence of accuracy and loss over iterations: (a) The accuracy over interactions, and (b) The loss over iterations.



(a) Normal state (b) Re attack state (c) OaU attack state (d) FoT attack state

(e) DoS attack state (f) DeC attack state (g) FDV attack state

Fig. 9: Real-time cyberattack detection: proposed Co-CNN-5 model in Ethereum node 1.

This is due to the different computer configurations between the two MNs described in section V-A. However, the current Ethereum mainnet has a maximum recorded transaction rate of 93.01 transactions per second (TPS) [42]. Note that each mining node only needs to process a smaller number of TPS, as the overall network TPS is distributed among thousands of nodes. In theory, with the final Ethereum 2.0 upgrade, the network transaction rate is expected to rise significantly, reaching up to 100,000 TPS across thousands of mining nodes [43] (averaging a few hundred TPS per mining node, with some nodes capable of handling higher loads). In this scenario, with each mining node capable of handling up to 2,150 TPS, our proposed system demonstrates exceptional efficiency in analyzing and detecting attacks within blockchain networks, ensuring robust security even as transaction volumes scale. Therefore, the capacity of our proposed system can be well-adapted to detect attacks on the mainnet Ethereum system.

VI. Limitations and Potential Research Directions

A. Limitations of the Proposed Framework

A key limitation of our framework is its ability to detect new or previously unseen types of attacks. Since the model is trained on known data, its performance may degrade when confronted with attacks that differ substantially from those in the training set, which is a common challenge in supervised learning. Nonetheless, our collaborative learning model is designed to capture high-level behavioral and structural features, providing some capacity to generalize and identify

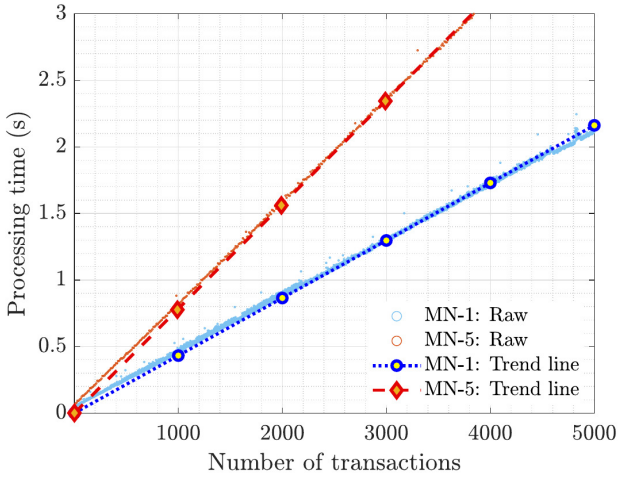


Fig. 10: Throughput of proposed Co-CNN-5 model in two computer configurations.

novel or evolving threats that share similar underlying patterns. Despite this potential, future work will explore more flexible learning approaches, such as semi-supervised or continual learning methods, to enhance the detection of unseen attacks. Another limitation involves class imbalance in the dataset. Some types of attacks are underrepresented, which can lead to biased learning and weaker performance on these less common classes. While basic techniques were used to reduce this issue, more advanced strategies may be needed to improve detection across all attack types.

B. Future Research Directions

As a potential research direction, we can expand and diversify datasets from multiple blockchain platforms, such as Solana and Binance Smart Chain, to improve the generalizability of the framework and assess its performance across different network environments. Additionally, we can apply our solution to practical applications built on these platforms, including decentralized autonomous organizations (DAOs), decentralized finance (DeFi) systems, and non-fungible token (NFT) marketplaces. Furthermore, to improve the transparency and trust of the model's predictions, we intend to investigate graph neural networks or hybrid regression models. Moreover, we plan to explore Explainable AI (XAI) techniques [44], such as saliency maps [45], Grad-CAM [46], or feature attribution techniques like Integrated Gradients [47] to highlight which parts of the input images influence the model's predictions.

To enhance robustness, we plan to integrate adversarial defense strategies, including adversarial training [48], feature consistency checks [49], and randomized smoothing [50], while evaluating the framework's performance under network latency on gradient exchange in large-scale blockchain environments. These approaches aim to enhance the model's resistance against evasion attempts and adversarial manipulations, supporting the framework's practical deployment in real-world blockchain systems where robustness to unseen and adversarial attacks is critical. We will also investigate more complex blockchain attacks in diverse and highly imbalanced scenarios, including cross-chain and side-chain attacks, while balancing model complexity with real-time processing requirements.

Beyond these technical considerations, our framework can have broader implications for enhancing trust and security in blockchain ecosystems, including DeFi platforms, DAOs, and NFT marketplaces. It also highlights the potential of collaborative learning for decentralized systems where data privacy and distributed processing are critical.

VII. Conclusions

In this work, we developed a collaborative learning model that can efficiently detect various types of attacks in transactions and SCs in a blockchain network. Our proposed solution is readily deployable in real-world blockchain systems, enhancing security through proactive and comprehensive attack detection. This positions the model as a practical and valuable tool for maintaining the integrity and reliability of blockchain operations. To do this, we implemented a private Ethereum network in our laboratory. We then performed attacks in transactions and SCs of that network for analysis. Next, we analyzed the transaction data and extracted the important features (i.e., Bytecode and value) to build the dataset. After that, we converted the dataset into grey images to train and evaluate the performance of our proposed model. In our proposed model, a learning node can detect attacks in transactions and SCs of a blockchain network and receive and aggregate learning knowledge (i.e., gradients) from other learning nodes to improve the accuracy of detection. In this way, our proposed model does not expose the local data of learning nodes over the network, thereby protecting the privacy of the local data of learning nodes. Both simulation results and real-time experimental results showed the efficiency of our proposed model in detecting attacks.

References

[1] R. Huo, S. Zeng, Z. Wang, J. Shang, W. Chen, T. Huang, S. Wang, F. R. Yu, and Y. Liu, "A comprehensive survey on blockchain in industrial Internet of things: Motivations, research progresses, and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 88–122, Jan. 2022.

[2] Y. Zuo, J. Guo, N. Gao, Y. Zhu, S. Jin, and X. Li, "A survey of blockchain and artificial intelligence for 6G wireless communications," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 2494–2528, Sep. 2023.

[3] J. Kang, J. Wen, D. Ye, B. Lai, T. Wu, Z. Xiong, J. Nie, D. Niyato, Y. Zhang, and S. Xie, "Blockchain-empowered federated learning for healthcare metaverses: User-centric incentive mechanism with optimal data freshness," *IEEE Transactions on Cognitive Communications and Networking*, vol. 10, no. 1, pp. 348–362, Feb. 2024.

[4] Z. Cheng, Y. Liang, Y. Zhao, S. Wang, and C. Sun, "A multi-blockchain scheme for distributed spectrum sharing in CBRS system," *IEEE Transactions on Cognitive Communications and Networking*, vol. 9, no. 2, pp. 266–280, Apr. 2023.

[5] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," Ethereum Foundation, Tech. Rep., Jan. 2014.

[6] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F.-Y. Wang, "Blockchain-enabled smart contracts: architecture, applications, and future trends," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2266–2277, Feb. 2019.

[7] SmartContractSecurity, "SWC Registry - Smart Contract Weakness Classification and Test Cases," Accessed: Nov. 23, 2025. [Online]. Available: https://swcregistry.io

[8] M. Ul Hassan, M. H. Rehmani, and J. Chen, "Anomaly detection in blockchain networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 289–318, Sep. 2023.

[9] J. Leng, M. Zhou, J. L. Zhao, Y. Huang, and Y. Bian, "Blockchain security: A survey of techniques and research directions," *IEEE Transactions on Services Computing*, vol. 15, no. 4, pp. 2490–2510, Jul. 2022.

[10] Z. Jiang, Z. Zheng, K. Chen, X. Luo, X. Tang, and Y. Li, "Exploring smart contract recommendation: towards efficient blockchain development," *IEEE Transactions on Services Computing*, vol. 16, no. 3, pp. 1822–1832, Aug. 2022.

[11] E. Banisadr, "How $800k Evaporated from the PoWH Coin Ponzi Scheme Overnight," Accessed: Nov. 23, 2025. [Online]. Available: https://medium.com/@ebanisadr/how-800k-evaporated-from-the-powh-coin-ponzi-scheme-overnight-1b025c33b530

[12] Y. Yuan and F.-Y. Wang, "Blockchain and cryptocurrencies: Model, techniques, and applications," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 9, pp. 1421–1428, Sep. 2018.

[13] R. Li, T. Song, B. Mei, H. Li, X. Cheng, and L. Sun, "Blockchain for large-scale internet of things data storage and protection," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 762–771, Sep. 2019.

[14] J. Huang, S. Han, W. You, W. Shi, B. Liang, J. Wu, and Y. Wu, "Hunting vulnerable smart contracts via graph embedding based bytecode matching," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2144–2156, Jan. 2021.

[15] J. Chen, X. Xia, D. Lo, J. Grundy, X. Luo, and T. Chen, "Defectchecker: Automated smart contract defect detection by analyzing evm bytecode," *IEEE Transactions on Software Engineering*, vol. 48, no. 7, pp. 2189–2207, Jan. 2021.

[16] P. Qian, Z. Liu, Q. He, R. Zimmermann, and X. Wang, "Towards automated reentrancy detection for smart contracts based on sequential models," *IEEE Access*, vol. 8, pp. 19 685–19 695, Jan. 2020.

[17] W. Wang, J. Song, G. Xu, Y. Li, H. Wang, and C. Su, "Contractward: Automated vulnerability detection models for ethereum smart contracts," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1133–1144, Jan. 2020.

[18] X. Wang, J. He, Z. Xie, G. Zhao, and S.-C. Cheung, "Contractguard: Defend ethereum smart contracts with embedded intrusion detection," *IEEE Transactions on Services Computing*, vol. 13, no. 2, pp. 314–328, Oct. 2019.

[19] Q.-B. Nguyen, A.-Q. Nguyen, V.-H. Nguyen, T. Nguyen-Le, and K. Nguyen-An, "Detect abnormal behaviours in ethereum smart contracts using attack vectors," in *International Conference on Future Data and Security Engineering (FDSE)*, Nha Trang, Vietnam, Nov. 2019, pp. 485–505.

[20] N. Ivanov, Q. Yan, and A. Kompalli, "Txt: Real-time transaction encapsulation for Ethereum smart contracts," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 1141–1155, Jan. 2023.

[21] J. Chen, X. Xia, D. Lo, J. Grundy, X. Luo, and T. Chen, "Defining smart contract defects on ethereum," *IEEE Transactions on Software Engineering*, vol. 48, no. 1, pp. 327–345, Jan. 2022.

[22] Z. Abou El Houda, A. Hafid, and L. Khoukhi, "Brainchain-a machine learning approach for protecting blockchain applications using sdn," in *IEEE International Conference on Communications (ICC)*, Dublin, Ireland, June 2020, pp. 1–6.

[23] R. Kumar, P. Kumar, R. Tripathi, G. P. Gupta, S. Garg, and M. M. Hassan, "A distributed intrusion detection system to detect ddos attacks in blockchain-enabled iot network," *Journal of Parallel and Distributed Computing*, vol. 164, pp. 55–68, June 2022.

[24] T. V. Khoa, D. H. Son, D. T. Hoang, N. L. Trung, T. T. T. Quynh, D. N. Nguyen, N. V. Ha, and E. Dutkiewicz, "Collaborative learning for cyberattack detection in blockchain networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 54, no. 7, pp. 3920–3933, Apr. 2024.

[25] B. Lashkari and P. Musilek, "A comprehensive review of blockchain consensus mechanisms," *IEEE Access*, vol. 9, pp. 43 620–43 652, Mar. 2021.

[26] V. Buterin, D. Hernandez, T. Kamphefner, K. Pham, Z. Qiao, D. Ryan, J. Sin, Y. Wang, and Y. X. Zhang, "Combining ghost and casper," May 2020. [Online]. Available: https://arxiv.org/abs/2003.03052

[27] H. Chen, M. Pendleton, L. Njilla, and S. Xu, "A survey on ethereum systems security: Vulnerabilities, attacks, and defenses," *ACM Computing Surveys*, vol. 53, no. 3, pp. 1–43, May 2021.

[28] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, Florida, USA, Apr. 2017, pp. 1273–1282.

[29] L. Hollander, "Evm bytecode decompiler," Accessed: Feb. 10, 2025. [Online]. Available: https://www.npmjs.com/package/evm

[30] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Computation*, vol. 29, no. 9, pp. 2352–2449, Aug. 2017.

[31] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[32] Y. M. Saputra, D. Nguyen, H. T. Dinh, Q.-V. Pham, E. Dutkiewicz, and W.-J. Hwang, "Federated learning framework with straggling mitigation and privacy-awareness for AI-based mobile application services," *IEEE Transactions on Mobile Computing*, vol. 22, no. 9, pp. 5296–5312, Sept. 2023.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–15.

[34] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," Oct. 2016. [Online]. Available: https://arxiv.org/abs/1610.02527

[35] Ethereum, "Official Go implementation of the Ethereum protocol," Accessed: Nov. 18, 2025. [Online]. Available: https://github.com/ethereum/go-ethereum/tree/v1.10.22

[36] Prysmatic Labs, "Prysm: An Ethereum Consensus Implementation Written in Go," Accessed: Jan. 10, 2025. [Online]. Available: https://github.com/prysmaticlabs/prysm/tree/v3.2.0

[37] A. Said, M. U. Janjua, S.-U. Hassan, Z. Muzammal, T. Saleem, T. Thaipisutikul, S. Tuarob, and R. Nawaz, "Detailed analysis of ethereum network on transaction behavior, community structure and link prediction," *PeerJ Computer Science*, vol. 7, pp. 1–26, Dec. 2021.

[38] OpenZeppelin, "A library for secure smart contract development," Accessed: Nov. 23, 2025. [Online]. Available: https://github.com/OpenZeppelin/openzeppelin-contracts

[39] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, Nov. 2008.

[40] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, June 2006.

[41] D. M. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, pp. 37–63, Oct. 2011.

[42] Etherscan, "Ethereum daily transactions chart," Accessed: Nov. 23, 2025. [Online]. Available: https://etherscan.io/chart/tx

[43] Vbuterin, "What would a rollup-centric ethereum roadmap look like?" Accessed: Nov. 23, 2025. [Online]. Available: https://ethereum-magicians.org/t/a-rollup-centric-ethereum-roadmap/4698/1

[44] R. Kumar, D. Javeed, A. Aljuhani, A. Jolfaei, P. Kumar, and A. K. M. N. Islam, "Blockchain-based authentication and explainable ai for securing consumer iot applications," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 1145–1154, Feb. 2024.

[45] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," Apr. 2014. [Online]. Available: https://arxiv.org/abs/1312.6034

[46] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, Oct. 2017, pp. 618–626.

[47] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International Conference on Machine Learning (ICML)*, Sydney, Australia, Aug. 2017, pp. 3319–3328.

[48] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, May 2016, pp. 1–11.

[49] X. Ma, B. Li, Y. Wang, S. M. Erfani, S. Wijewickrema, G. Schoenebeck, D. Song, M. E. Houle, and J. Bailey, "Characterizing adversarial subspaces using local intrinsic dimensionality," in *International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada, May 2018, pp. 1–15.

[50] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *International Conference on Machine Learning (ICML)*, California, USA, May 2019, pp. 1310–1320.