Vision-Based Learning for Cyberattack Detection in Blockchain Smart Contracts and Transactions

Do Hai Son*¶, Le Vu Hieu†, Tran Viet Khoa‡, Yibeltal F. Alem‡,
Hoang Trong Minh§, Tran Thi Thuy Quynh†, Nguyen Viet Ha†, and Nguyen Linh Trung†.

* VNU Information Technology Institute, Hanoi, Vietnam.
† VNU University of Engineering and Technology, Hanoi, Vietnam.
¶ School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Australia.
‡ University of Canberra, Australia.
§ Posts and Telecommunications Institute of Technology, Vietnam.

Abstract—**Blockchain technology has experienced rapid growth and has been widely adopted across various sectors, including healthcare, finance, and energy. However, blockchain platforms remain vulnerable to a broad range of cyberattacks, particularly those aimed at exploiting transactions and smart contracts (SCs) to steal digital assets or compromise system integrity. To address this issue, we propose a novel and effective framework for detecting cyberattacks within blockchain systems. Our framework begins with a preprocessing tool that uses Natural Language Processing (NLP) techniques to transform key features of blockchain transactions into image representations. These images are then analyzed through vision-based analysis using Vision Transformers (ViT), a recent advancement in computer vision known for its superior ability to capture complex patterns and semantic relationships. By integrating NLP-based preprocessing with vision-based learning, our framework can detect a wide variety of attack types. Experimental evaluations on benchmark datasets demonstrate that our approach significantly outperforms existing state-of-the-art methods in terms of both accuracy (achieving 99.5%) and robustness in cyberattack detection for blockchain transactions and SCs.**

Index Terms—**Cyberattack detection, blockchain, machine learning, deep learning, vision transformer, and cybersecurity.**

I. Introduction

Blockchain technology has advanced rapidly in recent years, particularly in the area of data management. By storing data on a distributed ledger, blockchain provides key properties such as immutability, security, and transparency. These properties ensure that data remains tamper-resistant, securely stored, and transparently accessible throughout the network, which strengthens the overall integrity and trustworthiness of the system. Its core features, including decentralization, immutability, and fault tolerance, make it a promising solution for a wide range of applications. As a result, blockchain has been increasingly adopted in sectors such as finance, healthcare, energy, and the Internet-of-Things (IoT) [1]. Despite these advantages, many blockchain systems have recently become targets of security threats, particularly in transactions and SCs. For instance, as of December 2024, data from the DeFiHacksLabs Reproduce Repository reports over 150 smart contract attack incidents in 2024 alone, resulting in losses exceeding $328 million [2], [3]. Although most attacks so far have focused on financial systems, the increasing use of blockchain in critical sectors, such as healthcare and energy, raises concerns that future attacks may have direct and serious consequences for human well-being [4].

Corresponding author: Tran Viet Khoa (khoa.tran@canberra.edu.au).

Several studies have explored cyberattack detection in blockchain SCs and transactions (txs). In [5], the authors proposed ContractWard, a system that applied machine learning models to detect six types of attacks, achieving an F1-score of up to 97%. However, their approach analyzed the source code of SCs rather than the bytecode. This presents a limitation, as only bytecode is available once a contract is deployed and executed through transactions. Therefore, direct analysis of bytecode is crucial for real-time attack detection. In [6], the authors used predefined attack vectors to analyze bytecode directly. While effective for specific attacks, their method was evaluated on a small dataset of around 100 samples, limiting its ability to detect new or evolving threats. In [7], the authors proposed a machine learning (ML) framework that performed real-time detection of six types of attacks by directly analyzing bytecode in SCs and transactions. They also introduced the BTAT dataset, which includes labeled cyberattack samples from blockchain environments. Experimental results showed their framework achieved an accuracy of up to 94% on this dataset. In [8], the authors developed an anomaly detection model for blockchain-based supply chain systems using only network-layer traffic data. While their approach achieved a promising accuracy of 96.5%, it was limited to general anomaly detection and did not target specific cyberattack types within SCs or transaction-level behaviors.

There are several challenges in detecting cyberattacks in SCs and transactions within blockchain systems. Firstly, improving the accuracy of cyberattack detection is crucial. Low accuracy can lead to missed attacks or false alarms, resulting in financial losses, security breaches, and reduced trust in blockchain systems. In contrast, high accuracy enhances detection reliability and supports stronger protection for decentralized applications. Secondly, when an SC is deployed, it is converted into bytecode, which is a sequence of hexadecimal values that represents the core logic of the contract and its associated transactions. Analyzing this bytecode is essential for real-time attack detection [9]. There are two common approaches for bytecode analysis: using the source code or analyzing the bytecode directly. However, only about 1% of SCs' source code is publicly available, making source-based analysis infeasible in most cases, while direct bytecode analysis is often time-consuming and less reliable [9].

To address the first challenge, we propose a novel framework for detecting cyberattacks in SCs and transactions with higher accuracy than existing methods. Our approach uses a ViT-based architecture, which allows the model to capture complex patterns more effectively than traditional methods. This makes our solution not only more accurate but also more flexible and easier to extend for detecting additional types of attacks compared to existing vector-based approaches. To tackle the second challenge, we design a preprocessing framework that uses NLP techniques to convert blockchain transaction features, such as opcodes (a structured representation of bytecode), into image representations. In addition, we extract and analyze critical transaction attributes, including transaction value, gas usage, and input length, to enrich the feature set and improve detection accuracy. We carefully study the importance of each feature and apply specific preprocessing approaches to improve its usefulness for attack detection. Experimental evaluations on the BTAT [7] dataset demonstrate that our model achieves a classification accuracy of 99.5% compared to baseline ML models (LR, KNN, SVM) and deep learning (DL) models (i.e., CNN, ResNet, MobileNetv2). Remarkably, our proposed ViT-based model uses only 8% of the trainable parameters of ResNet [10], indicating a lightweight architecture and improved computational efficiency without loss of accuracy.

II. Our Proposed Framework

This section introduces the key components of our proposed framework for cyberattack detection in blockchain SCs and transactions. First, we present the system model to clarify the deployment context and operational assumptions. Next, we explain the NLP-based preprocessing technique designed to convert transaction data into a unified image representation. Then, we describe the vision-based learning architecture employed for processing and classification.

A. System model

In this work, we focus on cyberattack detection in Ethereum SCs and transactions. As shown in Fig. 1, we assume that transactions at each Ethereum node (a.k.a, end-point) originate from various agencies, each with distinct purposes. For example, end-users exchange the native Ethereum token (ETH) either directly or through decentralized wallets such as MetaMask. Decentralized applications (e.g., mobile games, IoT, and supply chain platforms) also rely on Ethereum SCs to store digital assets and information. This information is considered trustworthy due to the immutability of blockchain technology. However, in the same manner, attackers can exploit intrinsic

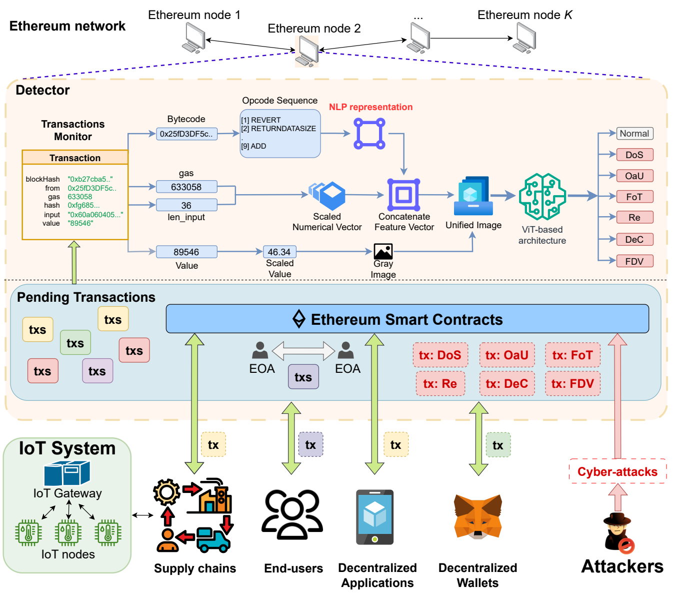


Fig. 1: The proposed model of cyberattack detection in Ethereum smart contracts and transactions.

vulnerabilities in the Ethereum protocol and the consensus mechanism, as well as weaknesses in SCs, to disrupt the network or steal digital assets and valuable information [4].

To detect these malicious SCs and transactions, we deploy a detector at the top of each Ethereum node, as illustrated in Fig. 1. Therefore, this detector can collect pending transactions (e.g., transactions and SCs' deployments) from the Ethereum node. The process of classifying a transaction consists of three stages. First, relevant fields inside a transaction, i.e., input (a.k.a., bytecode), length of input (len_input), gas, and value, are extracted. Second, these fields are processed to generate an NLP representation for the input, and normalized into vectors for gas, len_input, and value. The outputs of these processes are then fused into a unified image representation applicable to both SCs and transactions. Finally, our proposed DL model, based on the ViT architecture, takes this image as input and outputs a classification result to alert the administrator/end-users if malicious behavior is detected.

B. NLP-based Preprocessing Technique

During preprocessing, each transaction $\mathcal{T}_i \in \boldsymbol{\mathcal{T}}$, where $\boldsymbol{\mathcal{T}}$ denotes the dataset, is transformed into an image $\mathbf{I}_i$. This process consists of three main steps: feature extraction, feature assembly, and image construction.

1) Opcode-based Feature Extraction via TF-IDF: This stage captures the semantic content of transaction bytecode. Each transaction $\mathcal{T}_i$ consists of several fields, e.g., the input of hexadecimal bytecode b_i, gas usage g_i, value v_i, and so on. Specifically, the input length ℓ_i is computed as the number of bytes in $b_i \in \mathcal{T}_i$. To begin, we focus on the opcode-level semantics derived from the bytecode. The bytecode b_i

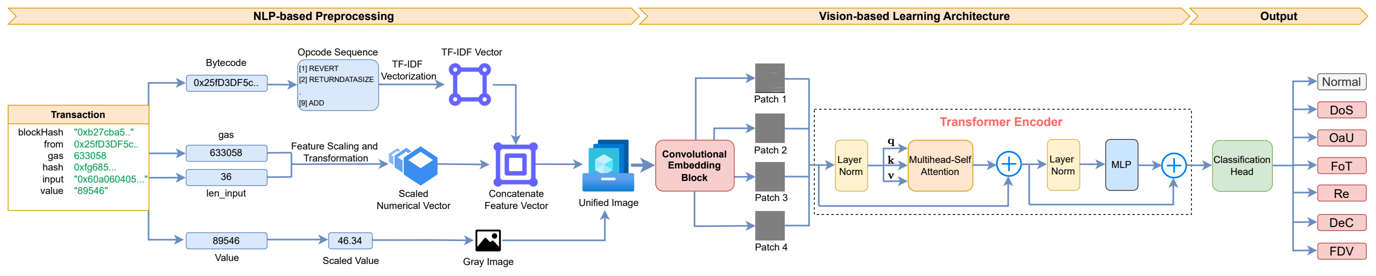


Fig. 2: The proposed NLP-based preprocessing and ViT-based architecture.

is decoded into an opcode sequence s_i using a disassembly function Φ_{op}:

$$s_i = \Phi_{\text{op}}(b_i). \tag{1}$$

The obtained opcode sequence s_i is treated as a text document, following an NLP perspective. This allows statistical representation through the Term Frequency–Inverse Document Frequency (TF-IDF) method [11]. For each term (i.e., opcode) t in s_i, the term frequency (TF) is given by:

$$\text{TF}(t, s_i) = f_{t,s_i}, \tag{2}$$

where f_{t,s_i} denotes the raw count of term t in document s_i. The inverse document frequency (IDF) is calculated with respect to the corpus $\mathcal{S}_{\text{train}}$, i.e., the set of opcode sequences extracted from the training transactions, as follows [11]:

$$\text{IDF}(t, \mathcal{S}_{\text{train}}) = \log \frac{|\mathcal{S}_{\text{train}}|}{|\{s \in \mathcal{S}_{\text{train}} : t \in s\}| + 1}. \tag{3}$$

The final TF-IDF score is the product of the two components [11]:

$$\text{TF-IDF}(t, s_i, \mathcal{S}_{\text{train}}) = \text{TF}(t, s_i) \cdot \text{IDF}(t, \mathcal{S}_{\text{train}}). \tag{4}$$

Let $\mathcal{V}_{\text{op}} = \{t_1, t_2, \ldots, t_{d_{\text{op}}}\}$ be the vocabulary of all unique opcodes extracted from $\mathcal{S}_{\text{train}}$. Applying the TF-IDF procedure to all opcodes in the vocabulary yields a fixed-dimensional feature vector:

$$\begin{aligned}\mathbf{u}_{\text{op},i} = [&\text{TF-IDF}(t_1, s_i, \mathcal{S}_{\text{train}}), \text{TF-IDF}(t_2, s_i, \mathcal{S}_{\text{train}}),\\ &\ldots, \text{TF-IDF}(t_{d_{\text{op}}}, s_i, \mathcal{S}_{\text{train}})]^T,\end{aligned} \tag{5}$$

where $\mathbf{u}_{\text{op},i} \in \mathbb{R}^{d_{\text{op}}}$ represents the semantic structure of transaction $\mathcal{T}_i$, and d_{op} is the size of the opcode vocabulary.

2) Feature Assembly and Image Transformation: In parallel, scalar attributes such as the gas usage g_i and bytecode length ℓ_i are normalized using standard-scaler to form the numerical attribute vector, as follows:

$$\mathbf{m}_i = [g_i, \quad \ell_i], \tag{6}$$

$$\mathbf{u}_i = \frac{\mathbf{m}_i - \mu_m}{\sigma_m}, \tag{7}$$

where $\mathbf{u}_i \in \mathbb{R}^{d_u}$ with d_u is the number of numerical features; $\mu_{\mathbf{m}}$ and $\sigma_{\mathbf{m}}$ are the mean and standard deviation of $\mathbf{m}$ computed on $\boldsymbol{\mathcal{T}}$, respectively. The final composite attribute vector is obtained via the following horizontal concatenation:

$$\mathbf{x}_i = [\mathbf{u}_i, \quad \mathbf{u}_{\text{op},i}], \tag{8}$$

with $\mathbf{x}_i \in \mathbb{R}^{d_u + d_{\text{op}}}$. This vector $\mathbf{x}_i$ is then interpolated and reshaped into a gray-scale matrix (gray image) $\mathbf{I}'_i \in \mathbb{R}^{H' \times W}$ through bilinear interpolation from $\mathbb{R}^{d_u + d_{\text{op}}} \rightarrow \mathbb{R}^{H' \times W}$, given by:

$$I'_{i,h,w} = \mathbf{r}_i(hW + w + 1), \tag{9}$$

where $\mathbf{r}_i \in \mathbb{R}^{H' \times W}$ is the bilinearly interpolated vector obtained by resizing $\mathbf{x}_i$, and $h \in \{0, 1, \ldots, H' - 1\}$, $w \in \{0, 1, \ldots, W - 1\}$. The value field is normalized as $a_i = 255\,(\log_{10}(v_i)/18)$. The normalized value a_i is then broadcast into a row vector $\mathbf{a}_i \in \mathbb{R}^{1 \times W}$ and appended as the last row of the image, as follows:

$$\mathbf{I}_i = [\mathbf{I}'_i, \quad \mathbf{a}_i]^T. \tag{10}$$

This image $\mathbf{I}_i \in \mathbb{R}^{H \times W}$ serves as the input for our vision-based learning model.

C. Vision-based Learning Architecture

In this work, we propose a ViT-based architecture that combines Vision Transformers [12] for detecting cyberattacks from preprocessed images with a hierarchical feature extraction mechanism inspired by ResNet [10]. To enable this, we use a convolutional embedding block that performs multi-stage convolutional patch embedding. Each input image is represented as $\mathbf{I}_i \in \mathbb{R}^{H \times W}$ and is first expanded into a tensor with a single channel. It is then passed through three sequential convolutional layers to extract feature representations [10]:

$$\mathbf{P}_i = \text{Conv}_3(\text{Conv}_2(\text{Conv}_1(\mathbf{I}_i))), \tag{11}$$

where $\text{Conv}_1 : \mathbb{R}^{1 \times H \times W} \rightarrow \mathbb{R}^{\frac{D}{4} \times H \times W}$, $\text{Conv}_2 : \mathbb{R}^{\frac{D}{4} \times H \times W} \rightarrow \mathbb{R}^{\frac{D}{2} \times \frac{H}{2} \times \frac{W}{2}}$, and $\text{Conv}_3 : \mathbb{R}^{\frac{D}{2} \times \frac{H}{2} \times \frac{W}{2}} \rightarrow \mathbb{R}^{D \times \frac{H}{4} \times \frac{W}{4}}$ with D as the embedding dimension of the Transformer. The resulting tensor $\mathbf{P}_i$ is flattened into a sequence of patches and concatenated with a learnable class token $\mathbf{z}_{\text{cls}} \in \mathbb{R}^{1 \times D}$ and positional embeddings $\mathbf{E}_{\text{pos}} \in \mathbb{R}^{(N+1) \times D}$, where $N = \frac{HW}{16}$ [12]:

$$\mathbf{Z}_i^0 = [\mathbf{z}_{\text{cls}}, \quad \text{Flatten}(\mathbf{P}_i)] + \mathbf{E}_{\text{pos}} \in \mathbb{R}^{(N+1) \times D}. \tag{12}$$

This sequence is then passed through K Transformer encoder blocks using pre-normalization for improved training stability. For each encoder layer indexed by $k \in \{1, \ldots, K\}$, LayerNorm is applied before each sub-layer. The output of the multi-head self-attention (MSA) sub-layer is computed as [12]:

$$\mathbf{Z}''_{i,k} = \text{MSA}(\text{LN}(\mathbf{Z}_{i,k-1})) + \mathbf{Z}_{i,k-1}, \tag{13}$$

where $\mathbf{Z}_{i,k-1}$ is the output of the previous layer. Each MSA block uses multiple attention heads $l \in \{1, \ldots, L\}$ with query, key, and value matrices computed as: $\mathbf{q}_l = \text{LN}(\mathbf{Z}_{i,k-1})\mathbf{W}_l^{\mathbf{q}}$, $\mathbf{k}_l = \text{LN}(\mathbf{Z}_{i,k-1})\mathbf{W}_l^{\mathbf{k}}$, and $\mathbf{v}_l = \text{LN}(\mathbf{Z}_{i,k-1})\mathbf{W}_l^{\mathbf{v}}$, where $\mathbf{W}_l^{\mathbf{q}}, \mathbf{W}_l^{\mathbf{k}}, \mathbf{W}_l^{\mathbf{v}} \in \mathbb{R}^{D\times\frac{D}{L}}$ are the weight matrices of $\mathbf{q}_l$, $\mathbf{k}_l$, $\mathbf{v}_l$, respectively. With $D_l = \frac{D}{L}$, the self-attention output of each head is [12]:

$$\mathbf{S}_{i,l} = \mathbf{v}_l \cdot \text{softmax}\left(\frac{\mathbf{q}_l\mathbf{k}_l^T}{\sqrt{D_l}}\right) \in \mathbb{R}^{(N+1)\times\frac{D}{L}}. \quad (14)$$

The multi-head attention output is then combined and linearly projected [12]:

$$\text{MSA}(\text{LN}(\mathbf{Z}_{i,k-1})) = [\mathbf{S}_{i,1}, \mathbf{S}_{i,2}, \ldots, \mathbf{S}_{i,L}]\mathbf{W}_{\text{msa}} \in \mathbb{R}^{(N+1)\times D}, \quad (15)$$

where $\mathbf{W}_{\text{msa}} \in \mathbb{R}^{D\times D}$ is the output projection matrix of the multi-head self-attention (MSA) module. The output is then passed through a Multilayer Perceptron (MLP) network using pre-normalization [12]:

$$\mathbf{Z}_{i,k} = \text{MLP}(\text{LN}(\mathbf{Z}''_{i,k})) + \mathbf{Z}''_{i,k}, \quad (16)$$

where the MLP consists of two linear layers with weights $\mathbf{W}_1 \in \mathbb{R}^{D\times 4D}$ and $\mathbf{W}_2 \in \mathbb{R}^{4D\times D}$.

After K layers, the final representation $\mathbf{Z}_{i,K}$ is used for classification via the class token at index 0 [12]:

$$\mathbf{Y}_i = \mathbf{W}_{\text{head}} \cdot \text{LN}(\mathbf{Z}_{i,K}[0]) \in \mathbb{R}^{C_{\text{out}}}, \quad (17)$$

where $\mathbf{W}_{\text{head}} \in \mathbb{R}^{D\times C_{\text{out}}}$ is the weight matrix of the classification head, C_{out} is the number of output classes, and the output $\mathbf{Y}_i$ classifies the transaction as normal or a specific type of cyberattack.

III. Experimental Results

A. Dataset and Evaluation Methods

To evaluate the performance of our proposed framework, we use the Blockchain Transaction-based Attacks (BTAT) dataset, first introduced in [7]. This dataset was synthesized in a laboratory environment using a private Ethereum network, ensuring that transactions are accurately labeled as either normal or a specific type of attack. The dataset is designed to replicate various real-world attacks that have previously caused significant damage to blockchain systems.

The BTAT dataset includes a wide range of cyberattacks targeting SCs and transactions. The dataset comprises 302,749 transactions in total, including normal (152,423 samples) and six types of attacks as follows:

- **Denial of Service (DoS) with Block Gas Limit:** Attackers exploit functions whose gas requirements can be manipulated to exceed the block gas limit, rendering the contract temporarily unusable (22,994 samples).
- **Overflows and Underflows (OaU):** Arises from integer arithmetic vulnerabilities in Solidity, where variables wrap around upon reaching their maximum or minimum values, enabling attackers to bypass balance checks (29,254 samples).
- **Flooding of Transactions (FoT):** Involves spamming the network with a massive number of meaningless transactions to delay the confirmation of legitimate ones (41,732 samples).
- **Re-entrancy (Re):** Exploits vulnerabilities where a contract's state is not updated before an external call, allowing attackers to recursively withdraw funds, as famously occurred in The DAO attack (22,682 samples).
- **Delegatecall (DeC):** Involves manipulating the `delegatecall` mechanism to execute malicious code from an untrusted contract within the context of the main contract, replicating the Parity Multi-Sig Wallet attack (22,455 samples).
- **Function Default Visibility (FDV):** Occurs when functions without explicitly defined visibility default to public, allowing unauthorized users to call critical administrative functions, as seen in the first Parity Multi-Sig Wallet hack (11,209 samples).

TABLE I: Performances of CNN and ViT-based architectures using baseline and our proposed preprocessing techniques.

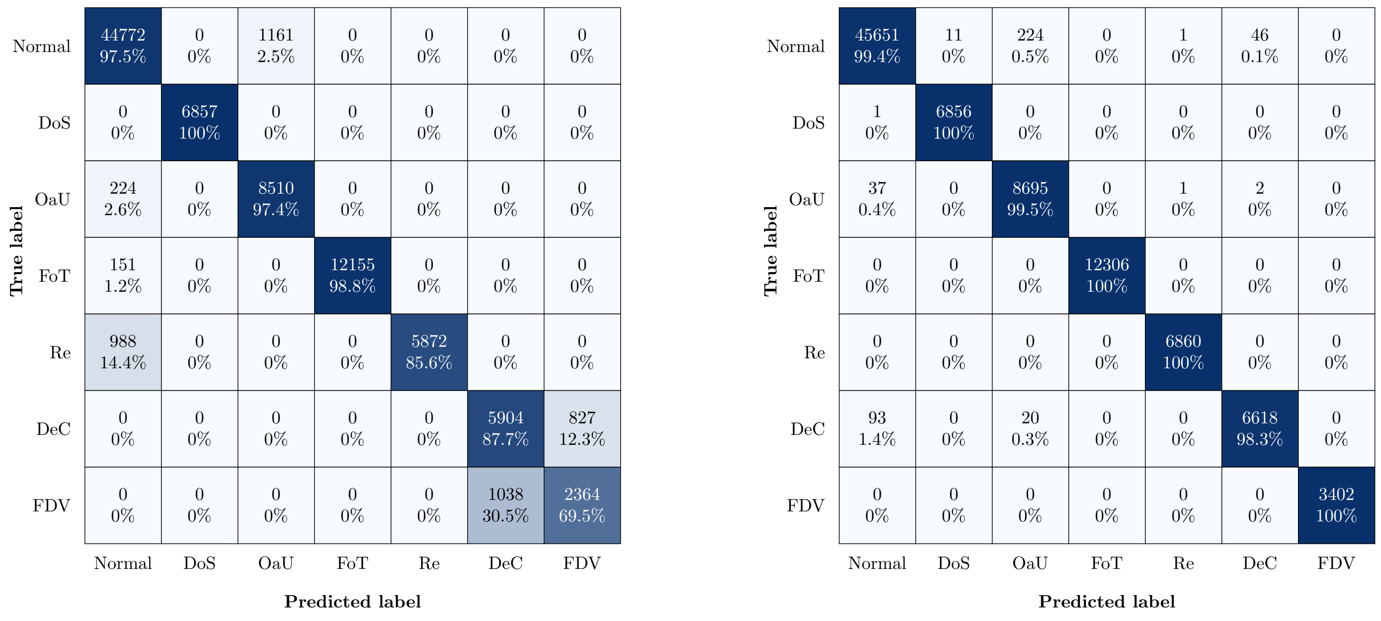
	CNN (Baseline)	CNN (Proposed)	ViT-based (Baseline)	ViT-based (Proposed)
Accuracy	93.8490	**98.8582**	95.1665	**99.5200**
Precision	90.4130	**98.5740**	92.0221	**99.4393**
Recall	89.7420	**99.2734**	95.2711	**99.6050**

To evaluate the performance of our models, we use several standard metrics that are widely used to evaluate the performance of DL [13] and cyberattack detection systems [14], such as: accuracy, precision, and recall. To ensure a fair evaluation across the imbalanced classes, we compute precision and recall using the 'macro' averaging method, which calculates the metric independently for each class and then takes the unweighted average.

B. Performance Evaluation and Analysis

1) The Impact of Preprocessing Techniques: In Table I, we compare the performances of CNN and our ViT-based models using different preprocessing techniques on the BTAT dataset [7]. Using the baseline preprocessing method from [7], CNN and our ViT-based models achieve accuracies of 93.8% and 95.16%, respectively, for detecting attacks in SCs and transactions. In contrast, when applying our proposed NLP-based preprocessing technique, the accuracies significantly improve to 98.8% for CNN and 99.52% for ViT-based models. These results demonstrate the effectiveness of our preprocessing method compared to using image transformation alone.

Figure 3(a) presents the classification results of the ViT-based model using both the baseline and our proposed NLP-based preprocessing methods. The results show that, with the same ViT-based architecture, our approach consistently outperforms the baseline across all attack types. Notably, our method improves the detection accuracy of FDV by approximately 30%, DeC by 11%, and Re by 14% compared to



(a) Image transformation preprocessing technique in [7] (b) Our proposed NLP-based preprocessing technique

Fig. 3: The classification results of ViT-based models with different preprocessing techniques.

the baseline. The proposed preprocessing technique combines TF-IDF vectorization, concatenated feature vectors, and a unified image combination. This technique helps capture richer semantic and structural information from transaction data, enabling the model to more effectively distinguish among normal and different types of attacks. Due to its improved performance, this preprocessing technique will be used in the following sections to evaluate and compare the performance of our proposed detection framework with existing approaches.

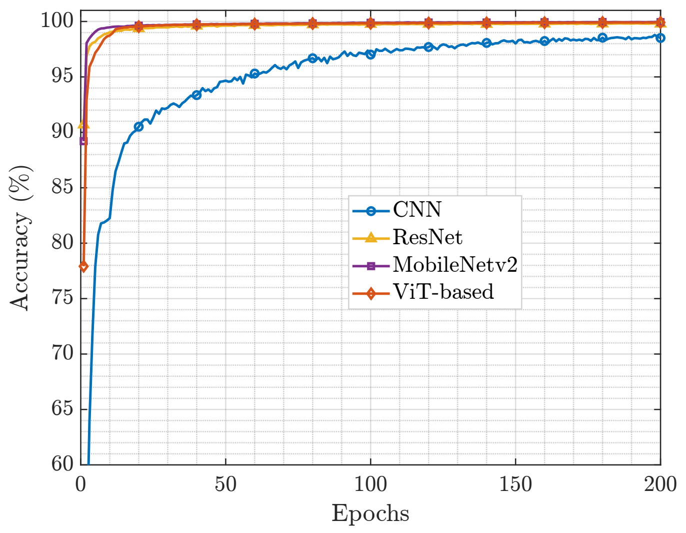


Fig. 4: The training convergence of accuracy over different DL models.

2) Model Convergence and Performance Comparison: In this section, we compare the convergence time and performance of our proposed architecture with other relevant models on the training dataset. The comparison contains traditional ML models, including logistic regression (LR), k-nearest neighbors (KNN), and support vector machines (SVM) [15], as well as DL models, such as Convolutional Neural Network (CNN) [7], ResNet [16], and MobileNetv2 [17]. As shown in Fig. 4, the three DL models, ResNet, MobileNetv2, and our proposed ViT-based, require fewer than 20 epochs to achieve near-perfect training accuracy (up to 100%) during the training phase. On the other hand, the CNN model on the BTAT dataset requires more than 150 epochs to stabilize at approximately 98.5% accuracy. While our proposed ViT-based model converges almost the same as ResNet and MobileNetv2, it demonstrates a rapid and stable learning process with fewer training epochs compared to the CNN model.

Table II summarizes the performance of these models on the testing dataset. Overall, our proposed ViT-based model outperforms all other considered approaches, with an accuracy of 99.5%, a precision of 99.4%, and a recall of 99.6%. The conventional ML models, i.e., LR, KNN, and SVM, deliver acceptable results given their simplicity, achieving accuracies of 94.8%, 95.5%, and 96%, respectively. Other DL models, such as CNN and MobileNetv2, produce test results that are consistent with their training performance, achieving accuracies of 98.8% and 99.0%, respectively. In contrast, the ResNet model shows a significant drop in testing accuracy, attaining only 89.5% despite almost perfect results during training. This performance degradation is likely due to overfitting caused by its large number of training parameters relative to the simplicity of the classification task. Furthermore, the detailed classification results of our ViT-based model, as shown in Fig. 3(b), demonstrate its robustness across all types of attacks in SCs and transactions. It achieves over 99.4% accuracy in

TABLE II: Performance of proposed model versus others.

	LR [15]	**KNN** [15]	**SVM** [15]	**CNN** [7]	**ResNet** [16]	**MobileNetv2** [17]	**ViT-based**
Accuracy	94.8836	95.5299	96.0198	98.8582	89.5446	99.0080	**99.5200**
Precision	94.6416	94.2707	95.7315	98.5740	87.9960	98.8280	**99.4393**
Recall	96.3814	94.7914	97.6686	99.2734	93.3549	99.2818	**99.6050**

TABLE III: Number of training parameters of proposed model versus others.

	CNN	**ResNet**	**MobileNetv2**	**ViT-based**
No. params	806,727	11,173,831	2,233,543	899,559

six out of seven classification categories.

3) Model Parameter Analysis: Table III compares the number of trainable parameters across the evaluated DL models. Among the four architectures, ResNet has the largest model size with over 11 million parameters, which explains its tendency to overfit the relatively simple classification task, as observed in the previous evaluation. MobileNetv2, designed for lightweight deployment, has approximately 2.2 million parameters. Our proposed ViT-based model contains only 899,559 parameters, significantly fewer than both ResNet and MobileNetv2, and almost the same as the baseline CNN with 806,727 parameters. Despite its compact size, our proposed ViT-based model achieves the best performance in terms of accuracy, precision, and recall, demonstrating its superior parameter efficiency and suitability for deployment in resource-constrained environments.

IV. Conclusion

In this paper, we proposed a novel framework that can efficiently detect cyberattacks in blockchain SCs and transactions in a blockchain system. The framework combined an NLP-based preprocessing technique with a ViT-based model to achieve accurate and efficient detection. In the preprocessing stage, opcode sequences were vectorized using the TF-IDF method and combined with key transaction features, such as gas, input length, and value, to form unified image representations. These images were then classified using the proposed vision-based architecture, which achieved a detection accuracy of 99.5%, outperforming other baseline models. We also analyzed the number of trainable parameters across different deep learning models to highlight the efficiency of our approach. Our proposed model is lightweight, with a size equal to only 40% of MobileNetv2 and 8% of ResNet, making it suitable for deployment on edge devices. In future work, we plan to evaluate the robustness of the framework under adversarial scenarios and extend the approach to support cross-chain transaction analysis.

Acknowledgment

This study was funded by the Australia-Vietnam Strategic Technologies Centre, PTIT, Vietnam

References

[1] K. Yue, Y. Zhang, Y. Chen, Y. Li, L. Zhao, C. Rong, and L. Chen, "A survey of decentralizing applications via blockchain: The 5g and beyond perspective," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2191–2217, Sep. 2021.

[2] Cymetrics, "2024 DeFi Smart Contract Hack Incident Review," 2024. [Online]. Available: https://tech-blog.cymetrics.io/en/posts/alice/2024_defi_hack/

[3] T. V. Khoa, D. H. Son, D. T. Hoang, N. L. Trung, T. T. T. Quynh, N. N. Diep, N. V. Ha, and D. Eryk, "Real-time cyberattack detection with collaborative learning for blockchain networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, Dubai, UAE, Apr. 2024, pp. 1–6.

[4] 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.

[5] 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.

[6] 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 *Future Data and Security Engineering: 6th International Conference (FDSE)*, Nha Trang City, Vietnam, Nov. 2019, pp. 485–505.

[7] T. V. Khoa, D. H. Son, C.-H. Nguyen, D. T. Hoang, D. N. Nguyen, T. T. T. Quynh, T.-M. Hoang, N. V. Ha, E. Dutkiewicz, A. Alsheikh *et al.*, "Collaborative learning framework to detect attacks in transactions and smart contracts," *arXiv preprint arXiv:2308.15804*, Aug. 2024.

[8] D. H. Son, B. D. Manh, T. V. Khoa, N. L. Trung, D. T. Hoang, H. T. Minh, Y. Alem, and L. Q. Minh, "Semi-supervised learning for anomaly detection in blockchain-based supply chains," in *International Symposium on Communications and Information Technologies (ISCIT)*, Bangkok, Thailand, Sep. 2024, pp. 140–145.

[9] 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.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.

[11] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations (ICLR)*, Vienna, Austria, May 2021.

[13] D. M. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, Feb. 2011.

[14] 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*, Apr. 2024.

[15] H. Zou, Z. Li, and X. Li, "Malicious code detection in smart contracts via opcode vectorization," *arXiv preprint arXiv: 2504.12720*, Apr. 2025.

[16] F. Hussain, S. G. Abbas, M. Husnain, U. U. Fayyaz, F. Shahzad, and G. A. Shah, "Iot dos and ddos attack detection using resnet," in *IEEE International Multitopic Conference (INMIC)*, Bahawalpur, Pakistan, Nov. 2020, pp. 1–6.

[17] Y. Alaca and Y. Çelik, "Cyber attack detection with qr code images using lightweight deep learning models," *Computers & Security*, vol. 126, p. 103065, Mar. 2023.