Journal of Neonatal Surgery
ISSN(Online): 2226-0439
Vol. 14, Issue 7 (2025)
https://www.jneonatalsurg.com

A Study on Smart Contract Vulnerability Detection Based on Transfer Learning

Sayali M. Poojari[1], Swapnil S. Sonawane[2]

[1]M.Tech Scholar, Vidyalankar Institute of Technology, Mumbai, India.

Email ID: kandsayali.sk@gmail.com

[2]Assistant Professor, Vidyalankar Institute of Technology, Mumbai, India.

Email ID: swapnil.sonawane@vit.edu.in

.Cite this paper as: Sayali M. Poojari, Swapnil S. Sonawane, (2025) A Study on Smart Contract Vulnerability Detection Based on Transfer Learning. *Journal of Neonatal Surgery*, 14 (7), 1007-1117.

ABSTRACT

This paper provides an analysis of the security threats which are present in blockchain smart contracts, namely the vulnerabilities that include re-entrancy, integer overflow, denial of service, and access control. Using various detection tools like Mythril, Slither, Oyente, and Securify, along with their limitations (false positives) and emphasizes the need for fortitude in security measures, since only growing levels of awareness will bolster further development of blockchain solutions. It classifies smart contract vulnerabilities using a three-module technique based on data from the Ethereum documentation and the Smart Contract Dataset repository. The method comprises extracting bytecode from Solidity files, creating images, and building three deep learning models: CNN, XCEPTION, and EfficientNet-B2. The Convolutional Neural Network (CNN) is the most effective technique, with an overall accuracy of 71 percent. XCEPTION and EfficientNet-B2 yield similar accuracy rates of 69 and 75 percent, respectively. The work contributes to our understanding of smart contract security and aims to reduce vulnerabilities in Ethereum smart contracts.

Keywords: *Smart Contracts, Vulnerability Classification, Ethereum, Deep Learning, Convolutional Neural Network (CNN), Blockchain, Vulnerabilities Re-entrancy Attacks, Integer Overflow, Denial of Service (DoS).*

1. INTRODUCTION

Smart contracts on blockchain are intended for independent performances but they are subject to vulnerabilities, such as re-entrancy or denial of service, which are responsible for monetary losses as well as trust erosion (Macrinici et al., 2018; Sun et al., 2020). Efficient auditing and vulnerability detection became key remedial measures in the effort to mitigate losses due to these vulnerabilities, encourage global trust to enhance the rapid and widespread adoption of blockchain technology, and foster new developments in emerging economies. This research looks into the lingering security considerations affecting the Ethereum and other blockchain ecosystems, thereby elaborating on the existing methodologies to detect vulnerabilities and their minimalistic limitations, especially in the adoption of accurate and harmonized processes for exact classification. Precisely, past work by (Abdelaziz and Hobor, 2023) explored only the XCEPTION and EfficientNet-B2 models in disconnection. This study innovatively targets these two sophisticated and highly accurate models by extracting features directly from the smart contract file's bytecode in the solidity file. This new novel aim at filling existing gaps and further improves pre-existing efforts such as the Deep Learning Vulnerability Analyzer (DLVA). Despite reasonable success, the DLVA is speculated to have limitations(Abdelaziz and Hobor, 2023). The promising performance of XCEPTION and EfficientNet-B2 in this analysis-sans their minor usage for this specific context before-focuses on their potential for augmenting the security of Ethereum smart contracts through higher accurate vulnerability classification. Ethereum is an instance of smart contracts under consideration because they play quite an important role in blockchain technologies, especially within the Ethereum ecosystem, where they are involved in sensitive activities and monetary transactions (Wohrer and Zdun, 2018). Since earlier researches like DLVA may not have much applicability to other blockchains and comprehensive benchmarking, this study wishes to apply multiple deep learning models over the same Ethereum smart contracts with an intention to discover vulnerabilities. Such categorization of these vulnerabilities becomes very useful in alleviating risks associated with them such as monetary losses and security threats, especially considering the inherent complexities and decentralized character of blockchain technology (Fadele Ayotunde Alaba et al., 2023; Khan et al., 2021). With increasing adoption of smart contracts, deep learning models therefore provide a good technique for deep analysis of smart contract code and mitigate the shortcomings of traditional vulnerability assessment methods in this fast-growing field.

2. LITERATURE REVIEW

2.1. Analysing Smart Contract Measures

Despite their inability to function without human intervention and oversight, smart contracts are meant to be automatically executable by the defined criteria (Turakhia et al., 2023). To remedy the security-wise shortcomings of smart contracts using Ethereum, (Liao et al., 2019) defined SoliAudit (which meant solidity audit). No proposal was spared; they all captured attacks, including the DAO incident of 2016. SoliAudit is a fuzz testing tool for grey-box using Solidity bytecode and machine learning to cover 13 vulnerabilities out of the total 10 online transaction threats from an open security organization. It has a high success rate of over 90% on about 18,000 Ethereum smart contracts and Capture-the-Flag datasets for vulnerabilities like re-entrancy and arithmetic overflows within zero custom knowledge or specific patterns. EVM bytecode enables control flow analysis at the bytecode level, thus giving independent and binding contract execution on a stack-based computer (Clack et al., 2016). (He et al., 2020) study production of decentralized applications enabled by Ethereum smart contract technology and emphasizes studies on vulnerabilities and their mitigation mechanisms in contracts particularly due to random number vulnerabilities such as in Fomo3d. The research analyzes security auditing methodologies regarding Ethereum smart contracts, elaborating on their advantages and disadvantages and showcasing the risks and ways of tackling them in fast-growing decentralized applications. A study analysing 21,270 discovered Ethereum blockchain smart contract vulnerabilities found that 504 were exploited, resulting in a maximum loss of 9,066 ETH (roughly 1.8 million USD), which is about 0.29 percent of the 3 million ETH (or 600 million USD) cited by some sources. While the need for further research is acknowledged, the report suggests that the potential effects of bad programming in smart contracts were overestimated (Zhuang et al., 2020).

2.2.Enhancing Smart Contract Vulnerability Detection

A new technique using graph neural networks (GNNs) is proposed to improve detection precision in smart contracts. In the process of building a contract graph, the syntactic and semantic elements of different smart contract functions are normalized, followed by the identification of salient nodes in an elimination step. To expedite the process of identifying vulnerabilities, a temporal message propagation network (TMP) and a degree-free graph convolutional neural network (DR-GCN) are proposed (Liu et al., 2023). A new approach for short facilities for smart contract vulnerability detection globes expert knowledge and GNNs. The way constructs a huge contract network with rich control- and data-flow semantics from the source code, normalizes it, and creates an elimination phase where important nodes are highlighted. A novel temporal message propagation network is also employed to generate attributes for the graph. This would create a proper solution for accurate and scalable detection of vulnerabilities. A case study on the Ethereum and VNT Chain platform showed greater detection accuracy with respect to re-entrancy, timestamp dependence, and infinite loop vulnerabilities by 89.15 percent, 89.02 percent, and 83.21 percent, respectively (Jiang et al., 2018).

2.3.Advancements in Smart Contract Vulnerability Detection

In spite of possible economic damages, many applications of smart contracts in blockchain technology have surfaced, thus creating more interest in smart contract vulnerability detection. Traditional approaches that relied on expert-specified criteria suffer some issues of generality, scalability, and accuracy. (Chen et al., 2021) have executed an empirical experiment on ChatGPT potential performance employing an automatic detector for smart contract vulnerability which has high recall rates yet low accuracy. (Liu et al., 2023) introduced a method that combines graph neural networks with expert patterns for the enhancement of smart contract vulnerability detection and established to perform better than the contemporary methods. On the other hand, (Sun et al., 2020) produced ASSBert, with a mixture of active and semi-supervised learning to find vulnerabilities with negligible labelled data. Jie et al. (Jie et al., 2023) showed a white box knowledge-enforcement approach that outshined existing ones in multimodal vulnerability mining. (Cai et al., 2022) suggested an adversarial machine learning-based solution for detecting smart contract vulnerabilities, which achieved 89.2% and 92.9% for precision and recall, respectively.

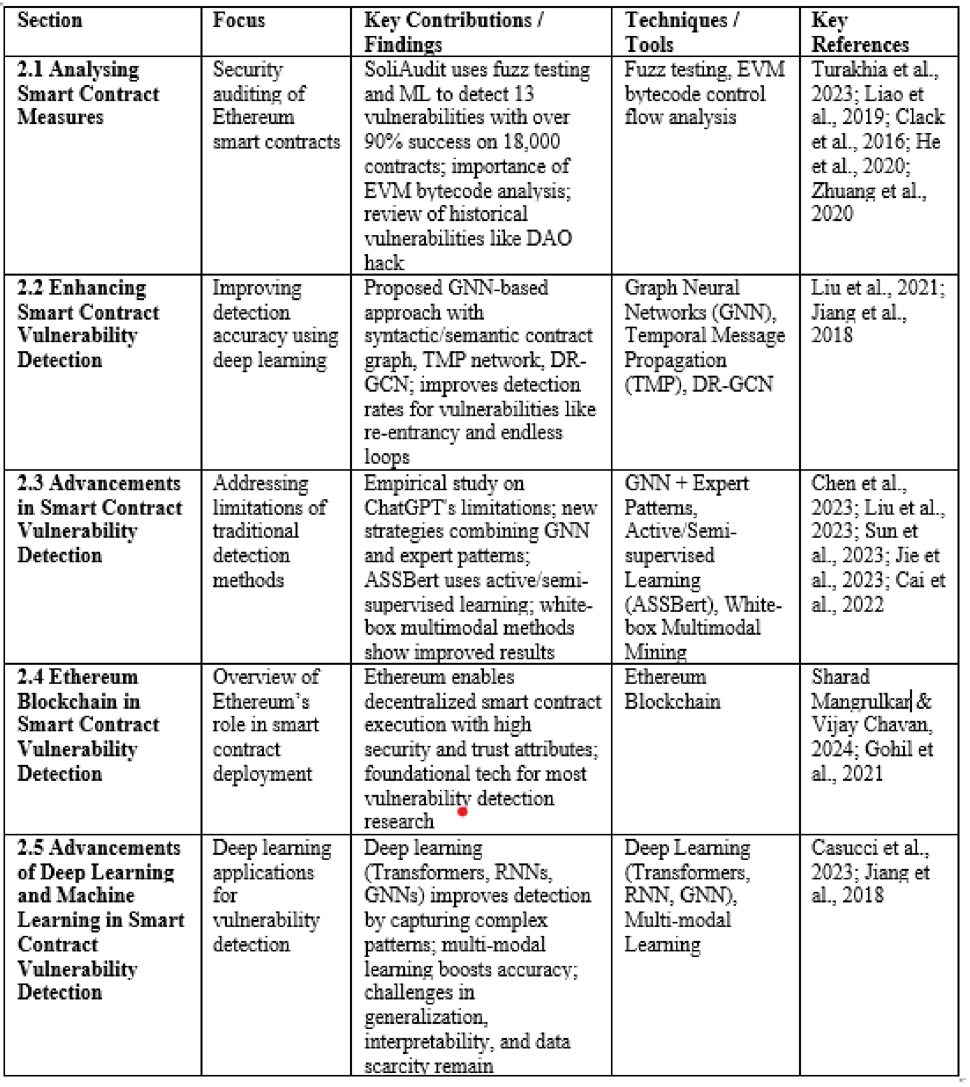
2.4.Ethereum Blockchain in Smart Contract Vulnerability Detection

Blockchain is an underlying technology that contains several forms of cryptocurrencies, including Ethereum and Bitcoin, which serve as the foundation for smart contracts. Ethereum is an open, decentralized source platform that allows for the execution and implementation of smart contracts. It was introduced by Vitalik Buterin in 2015 (Sharad Mangrulkar and Vijay Chavan, 2024). In commercial marketplaces, it adds new meaning to the phrase 'trust' and includes shared various blocks of transactions that are exceedingly secure (Gohil et al., 2021).

2.5.Advancements of Deep Learning and Machine Learning in Smart Contract Vulnerability Detection

Deep learning has also proved useful in the detection of weaknesses in smart contracts because it can generalize complex representations of input. Techniques like Transformers, Recurrent Neural Networks, and Graph Neural Networks help in extracting structural relationships in the smart contract code. Multi-modal learning is an integrated source of multiple representations of the smart contract information for improved accuracy. Generalization, model interpretability, and data scarcity seem to prevail. Machine learning (ML) is the science for understanding complex interactions between smart contract code and other aspects, too, but generalization and interpretability of models are its major issues. Recent innovations on blockchain platforms, mostly Ethereum, have greatly improved detection accuracy(Casucci et al., 2023; Jiang et al., 2018).

2.6.Summary of Key Findings



Section	Focus	Key Contributions / Findings	Techniques / Tools	Key References
2.1 Analysing Smart Contract Measures	Security auditing of Ethereum smart contracts	SoliAudit uses fuzz testing and ML to detect 13 vulnerabilities with over 90% success on 18,000 contracts; importance of EVM bytecode analysis; review of historical vulnerabilities like DAO hack	Fuzz testing, EVM bytecode control flow analysis	Turakhia et al., 2023; Liao et al., 2019; Clack et al., 2016; He et al., 2020; Zhuang et al., 2020
2.2 Enhancing Smart Contract Vulnerability Detection	Improving detection accuracy using deep learning	Proposed GNN-based approach with syntactic/semantic contract graph, TMP network, DR-GCN; improves detection rates for vulnerabilities like re-entrancy and endless loops	Graph Neural Networks (GNN), Temporal Message Propagation (TMP), DR-GCN	Liu et al., 2021; Jiang et al., 2018
2.3 Advancements in Smart Contract Vulnerability Detection	Addressing limitations of traditional detection methods	Empirical study on ChatGPT's limitations; new strategies combining GNN and expert patterns; ASSBert uses active/semi-supervised learning; white-box multimodal methods show improved results	GNN + Expert Patterns, Active/Semi-supervised Learning (ASSBert), White-box Multimodal Mining	Chen et al., 2023; Liu et al., 2023; Sun et al., 2023; Jie et al., 2023; Cai et al., 2022
2.4 Ethereum Blockchain in Smart Contract Vulnerability Detection	Overview of Ethereum's role in smart contract deployment	Ethereum enables decentralized smart contract execution with high security and trust attributes; foundational tech for most vulnerability detection research	Ethereum Blockchain	Sharad Mangrulkar & Vijay Chavan, 2024; Gohil et al., 2021
2.5 Advancements of Deep Learning and Machine Learning in Smart Contract Vulnerability Detection	Deep learning applications for vulnerability detection	Deep learning (Transformers, RNNs, GNNs) improves detection by capturing complex patterns; multi-modal learning boosts accuracy; challenges in generalization, interpretability, and data scarcity remain	Deep Learning (Transformers, RNN, GNN), Multi-modal Learning	Casucci et al., 2023; Jiang et al., 2018

3. METHODOLOGY

The paper proposes a deep learning model-based approach for identifying smart contract vulnerabilities. It extracts bytecode from Solidity files and uses it to generate images for training models. Deep learning models like CNN, Xception, and EfficientNet-B2 are used to categorize vulnerabilities and provide insights into potential flaws in smart contracts.(Eshghie et al., 2021; Mezina and Ometov, 2023). The research aims to create a deep learning model for classifying vulnerabilities in Ethereum smart contracts. The process involves data acquisition, preparation, and testing, with the dataset cleaned to remove missing values and irrelevant data. The model uses various deep learning architectures, including Recurrent Neural Networks, Convolutional Neural Networks, and hybrid models. Attention mechanisms are implemented to focus on relevant input sequences. Model training is performed using the training dataset, hyperparameter tuning is done using the validation dataset, and multi-task learning is explored. A Flask web application will be developed for vulnerability detection. The project consists of several modules, each contributing to the development and evaluation of a deep learning model for smart contract vulnerability detection.

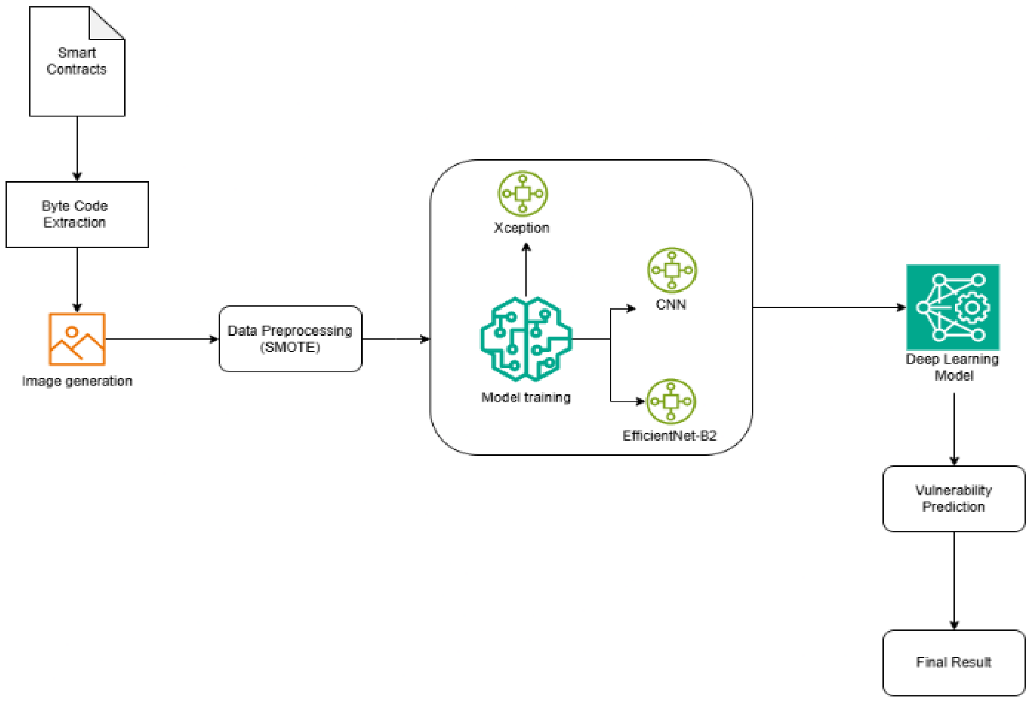


Figure 1: Network Architectural Diagram

3.1. Module 1: Bytecode Extraction and Image Generation

Module 1 focuses on feature extraction and representation, which involves translating Solidity source code into bytecode to represent smart contract execution. This produces a dataset of bytecode strings, which are disassembled into opcode sequences and feature vectorized for deep learning models. The research also examines picture creation from bytecode/opcode out of the Solidity files, which may capture spatial patterns and turn it into pictures. Module 1 attempts to transform Ethereum smart contract Solidity files into a format suitable for deep learning models by translating bytecode into images that will be utilized as input data for training deep learning models. If photos are generated, CNNs will be used to train the deep learning models in the next module.

a) **Extraction of Bytecode:** Solcx library extracts bytecode from Ethereum smart contract Solidity files, enabling faster compilation of source code and retrieval of associated bytecode, representing the smart contract's lowest-level commands.

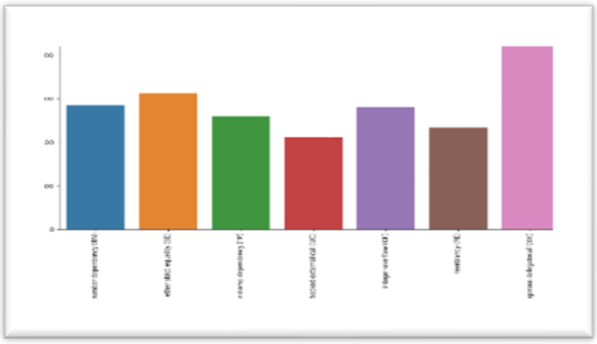
b) **Image Generation via Bytecode:** This retrieved bytecode is actually used to generate images by converting opcode to distinct visual representation effectively extracting complex patterns and relationships for training purposes in image-based deep learning models.

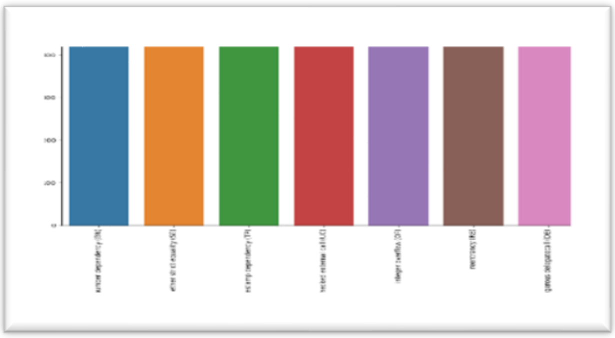
3.2. Module 2: Data Preprocessing and Model Development

This research module focuses on the design and assessment of deep learning models aimed at detecting vulnerabilities in Ethereum smart contracts utilizing bytecode represented as images.

The very first step in the process is ‘data loading’ from a picture dataset of bytecodes from smart contracts. Then, the following steps in ‘preprocessing’ are scaling of images, label binarization, and normalization. Using the Synthetic Minority Over-sampling Technique (SMOTE) introduced in the training data only, it will handle the ‘class imbalance’ found within the dataset such that it will better represent the different vulnerability types tested without bias to leakage of data. The ‘dataset’ contains roughly 2,000 Ethereum smart contracts, in addition to the inherited contracts, to model seven of the dominant security vulnerabilities. The bytecodes reflect the definition of images for the image-based approach, and they are actually from the documentation of Ethereum and the Smart Contract Dataset at GitHub. ‘Data visualization’ techniques which use Seaborn for count plots to detect class imbalance then Matplotlib to illustrate sample images from each vulnerability class would help in comprehending the dataset. The dataset will be partitioned into ‘training (90%) and testing (10%) sets’ using the scikit-learn's `train_test_split` while maintaining the class distribution across each set. ‘Model construction’ consists of three deep learning models that will be built: EfficientNet-B2, Xception, and a Convolutional Neural Network (CNN) using TensorFlow and Keras. Finally, ‘model evaluation’ will take place using the test dataset, which will involve employing confusion matrices, classification reports (precision, recall, F1, and support), specificity, sensitivity, and accuracy to quantify the performance of each model in terms of different detected vulnerabilities within smart contracts.

Figure 1 shows target class distribution before and after SMOTE application, highlighting distinct class splits and potential impact on machine learning model performance by making it difficult to anticipate underrepresented classes.



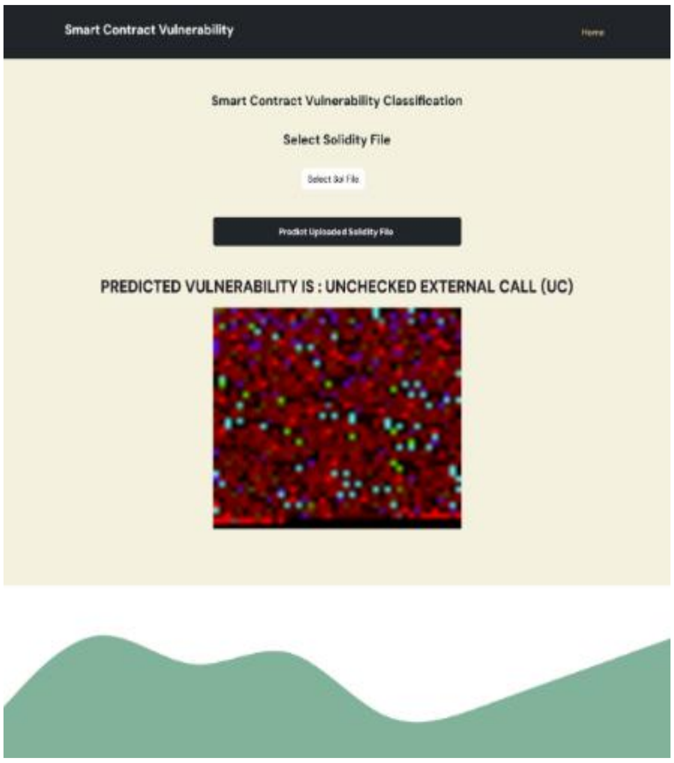


(1a) **(1b)**

Figure 2:Count plot of target class before (1a) and after (1b) applying SMOTE

3.3. Module 3: Web Application Development

Module 3 of the Flask project aims to create a user-friendly web application for detecting smart contract vulnerabilities. The process involves installing Flask, starting an instance, specifying routes, developing HTML templates, and designing a layout for uploading files and showing predictions. The Flask framework is used to build an intuitive application. Figure 2 predicts the vulnerability associated with unchecked external calls (UC).


Smart Contract Vulnerability
Home
Smart Contract Vulnerability Classification
Select Solidity File
Select Sol File
Predict Uploaded Solidity File

PREDICTED VULNERABILITY IS : UNCHECKED EXTERNAL CALL (UC)

Figure 3: Predicting Vulnerability is: Unchecked External Call (UC)

4. RESULTS AND DISCUSSION

Three deep learning models, CNN, Xception, and EfficientNet-B2, are used for photo categorization challenges, with TensorFlow and Keras used in the training phase.

To evaluate the suggested models, the following assessment indicators are taken into consideration:

- The 'Accuracy' measure shows the percentage of correctly detected instances in testing data, including both true positives and true negatives.
- The 'Precision' statistic measures the percentage of true positives based on the model's predictions. It shows the

proportion of positive episodes that really occurred vs those that were predicted. 'Recall', also known as sensitivity or true positive rate, is the proportion of true positives to all other occurrences of actual positive data. It assesses the model's capacity to account for all instances of success.

- The 'F1 score' represents the mean of recall and accuracy. False positives and false negatives are considered to offer a fair judgment. Because the measure considers both false positives and false negatives, it is extremely useful for dealing with unbalanced datasets.

4.1. CNN

Ethereum smart contracts are increasingly being targeted by malicious beings due to their complexity and financial relevance. Employing deep learning involves the probable use of Convolutional Neural Networks integrated with transfer learning to enhance the accuracy and efficiency by which vulnerabilities are detected. CNNs-which, at the beginning, are image-driven-can be used to detect vulnerabilities in smart contracts through opcode sequences or bytecode representations and abstract syntax trees, lessening the need for expensive manual feature construction. The goal of this study is to use pretrained CNN models to provide a classification of vulnerabilities based on image representations of smart contract bytecodes. The model learning paradigm is presented through epoch-wise metrics (loss, accuracy, validation loss, validation accuracy) as effective learning and good generalization to the validation data. While validation accuracy was much initially low (0.1536), it improved significantly to 0.7124 by the 10th epoch. The CNN had an all datum accuracy of 71% that suggests said ability to correctly categorize most cases in the dataset. While the training history indicates good learning and generalization, further analysis-in particular, loss and accuracy curves-would give a much fuller picture of the training dynamics and possibility of overfitting.

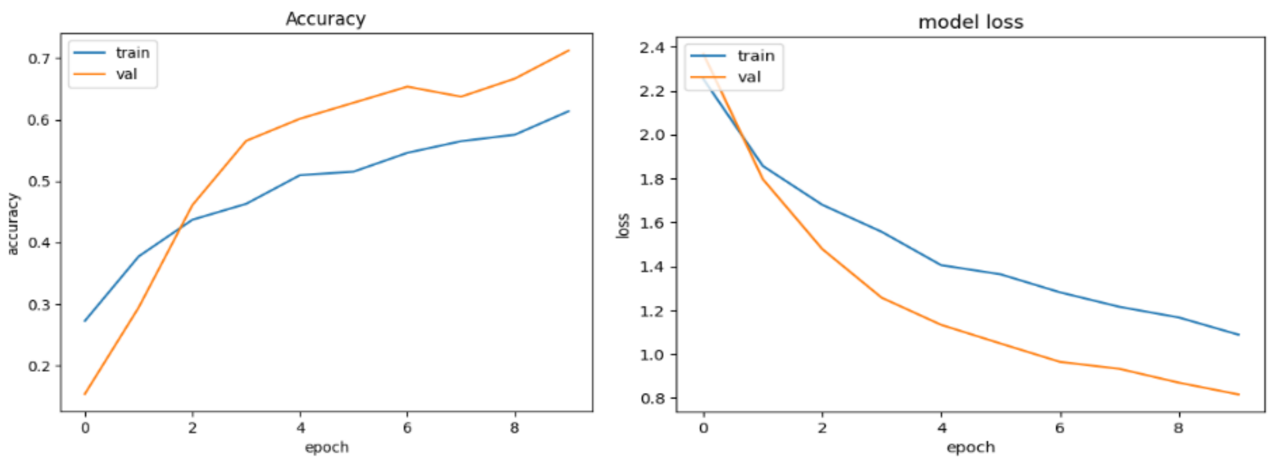


Figure 4: Accuracy and Loss graph using CNN Model

Figure 3 depicts the CNN Accuracy and Loss Graph, which demonstrates the model's performance over training and validation epochs. The blue line represents training accuracy, which improves with each epoch and demonstrates the model's capacity to learn new knowledge. The red line illustrates how well the model generalizes to unknown data. Both lines should exhibit an upward trend, demonstrating effective learning without overfitting. The CNN Loss Graph depicts the loss over the training and validation epochs. A reduction in both training and validation loss implies improved model learning. The confusion matrix of a CNN model visually displays the model's performance, highlighting correct predictions and potential misunderstandings. The model classifies data into seven classes, with high accuracy for 0, 1, and 5, and few misclassifications. However, it struggles with Class 4, with low predictions and significant misclassifications, especially for Class 6. Further investigation is needed through Data Augmentation, Feature Engineering, and Model Tuning to improve the model's accuracy and reduce misclassifications.

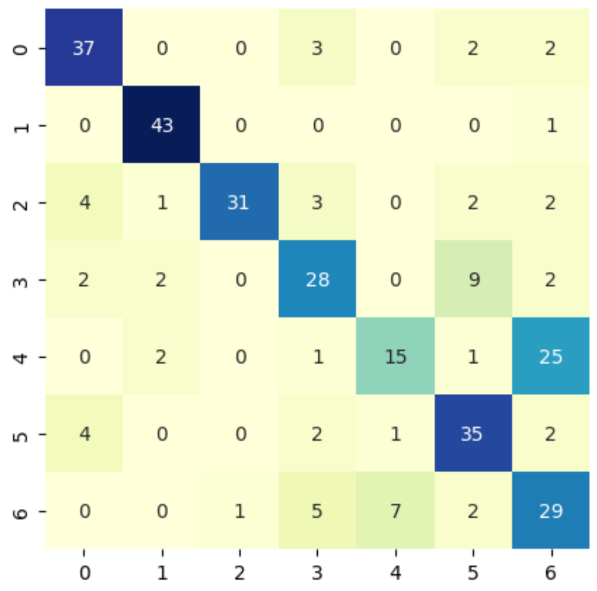


Figure 5:Confusion Matrix for CNN model

The categorization report of the model assesses the model's performance in seven categories (0–6). Class 1 offers the greatest results, with an accuracy of 0.90, recall of 0.98, and F1-score of 0.93. Class 2 and class 0 had F1-scores of 0.83 and 0.81, respectively. Class 3 has a modest performance, with misclassification mistakes. Class 4 has the lowest performance, with a recall of 0.34 and an F1-score of 0.45. Classes 5 and 6 have intermediate scores.

4.2. XCEPTION

Xception is a highly sophisticated deep learning model based on deep learning separable convolution and offers a good method of finding faults in smart contracts. It applies transfer learning in effectively digging the contract code representations and secures detentions of vulnerabilities such as re-entrancy, integer overflow, and access control problems. It specializes in extracting a very deep hierarchical feature set, which makes it accurate, robust, and, hence, a great method for automatic vulnerability detection in blockchain applications. The Exception model is a deep learning model that is trained for 10 epochs and has shown good promise in identifying vulnerabilities in smart contracts. It started with a rather unimpressive score of 34.23% when taken against the training set, dropping much lower to achieve 19.37% on the validation set. As training progressed, however, it kept on doing better with decreased loss values and increased accuracy, finally reaching 80.95% and 68.63%. The gradual rise of the model in accuracy and fall in loss suggest its capacity in discerning very complex patterns from the smart contract data set, thus becoming a candidate for vulnerability detection. However, the XCEPTION model scored 69% in categorizing most dataset cases, making it less accurate than the CNN Model.

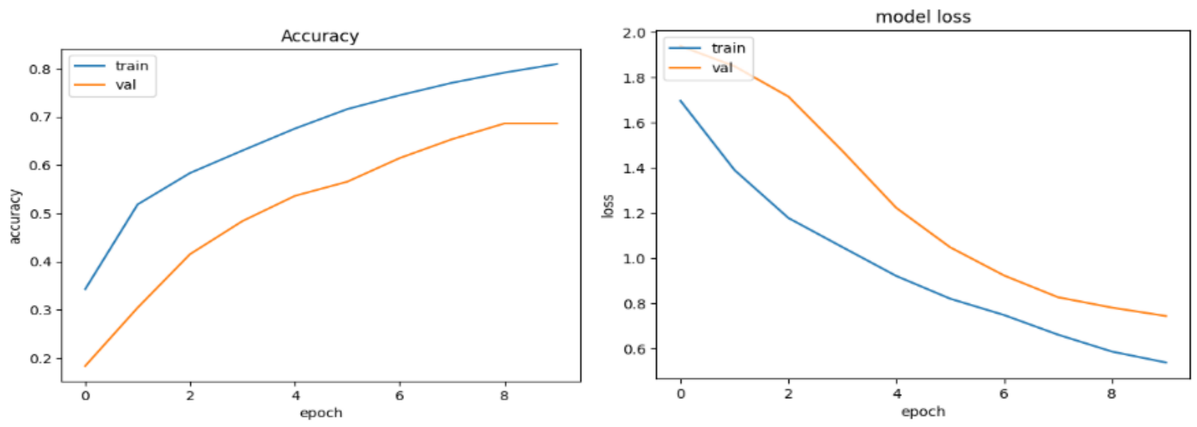
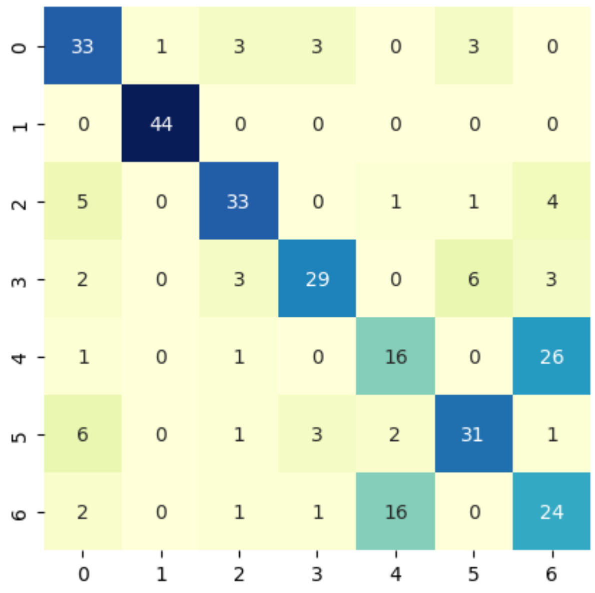


Figure 6: Accuracy and Loss Graph using XCEPTION model

Figure 5 depicts an accuracy and loss graph of the Xception model. The accuracy and loss graphs depict the training progress of a model across ten epochs. The accuracy plot shows a consistent improvement in both training and validation accuracy, with training accuracy above 0.8 and validation accuracy stabilizing at approximately 0.7. The loss plot demonstrates a steady decrease in both training and validation loss, showing that the model is learning efficiently. However, the difference between the training and validation curves indicates potential overfitting, since training performance increases more than validation performance.



	0	1	2	3	4	5	6
0	33	1	3	3	0	3	0
1	0	44	0	0	0	0	0
2	5	0	33	0	1	1	4
3	2	0	3	29	0	6	3
4	1	0	1	0	16	0	26
5	6	0	1	3	2	31	1
6	2	0	1	1	16	0	24

Figure 7:Confusion Matrix for EXCEPTION model

The model's classification performance across seven classes is assessed using accuracy, recall, and F1-score. Class 1 has the highest accuracy with 44 correctly categorized cases, while other classes, such as 2 and 6, exhibit significant misclassification. Class 6 has the lowest precision with 0.41 and recall, indicating frequent misclassifications. Classes 0, 2, 3, and 5 have made a nail-by-nail accuracy and recall score, rendering average F1 results. Class 4 has quite a poor recall (0.36) and has an F1-score of 0.41, meaning it is easily suspect to misclassification. The overall accuracy is 69% with macro and weighted averages lingering at 0.69. The model is consistent throughout classes but a little better at handling low or overlapping classes.

EFFICIENT NET-B2

The Smart Contract Vulnerability Detection using Transfer Learning with EfficientNet-B2 is a pre-trained deep learning model that improves security in smart contracts by identifying vulnerabilities like re-entrancy and overflow attacks. This technique reduces the need for vast labelled data and speeds up training, making it a scalable solution for protecting blockchain-based applications, while also improving generalization.

Accuracy and Model Loss Graph

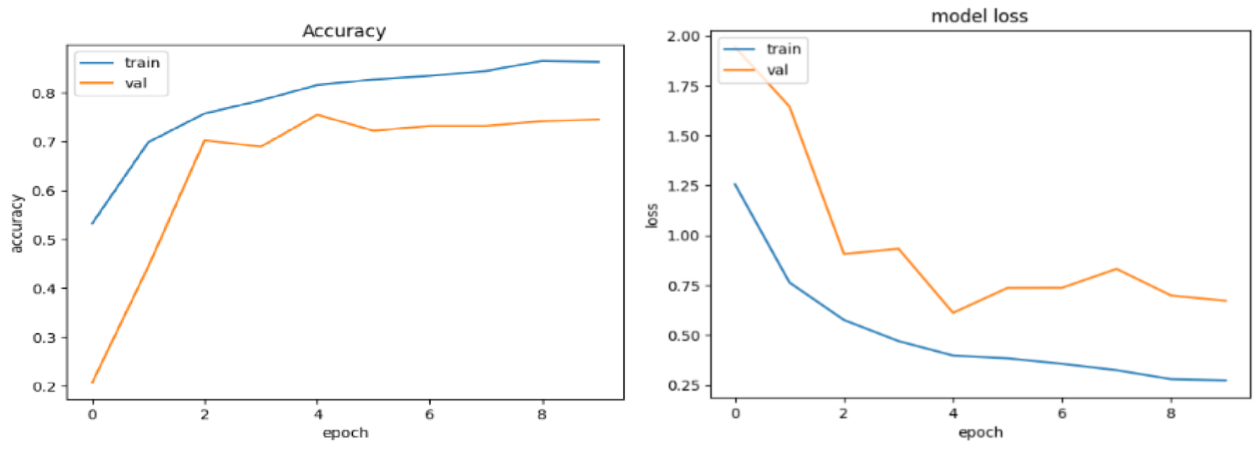


Figure 8: Accuracy and Loss Diagram for Efficient Net B2 model

Confusion Matrix

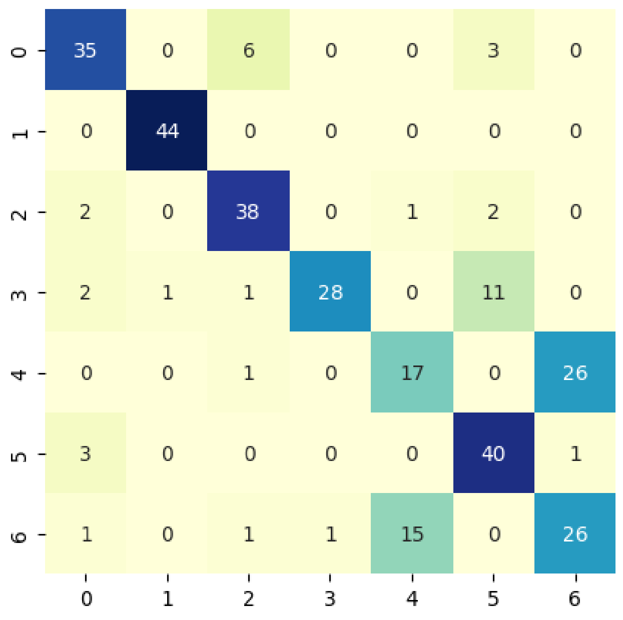
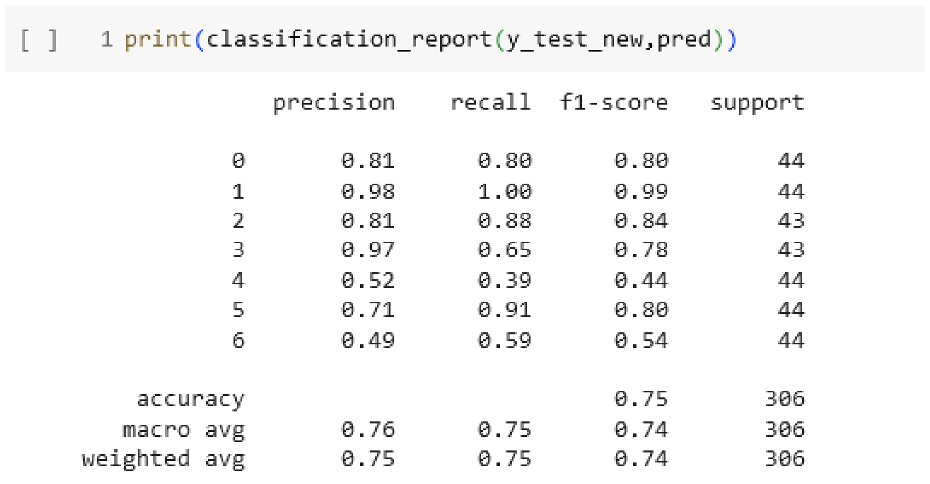


Figure 9: Confusion Matrix for Efficient Net B2 model

The confusion matrix shows the model's classification performance across seven classes, with diagonal elements indicating accurate predictions and off-diagonal values representing misclassifications. The model performs well in class 1 and class 5, with 44 correctly identified samples. However, there are significant misclassifications in class 4, with 26 samples misclassified as class 6, and class 3, with 11 misclassified samples. Misclassifications may occur due to overlapping feature distributions, class imbalances, or insufficient dataset representation. Enhancements include data supplementation, hyperparameter tweaking, and more complex feature selection approaches.

Classification Report



```
[ ]   print(classification_report(y_test_new,pred))
```

```
              precision    recall  f1-score   support

           0       0.81      0.80      0.80        44
           1       0.98      1.00      0.99        44
           2       0.81      0.88      0.84        43
           3       0.97      0.65      0.78        43
           4       0.52      0.39      0.44        44
           5       0.71      0.91      0.80        44
           6       0.49      0.59      0.54        44

    accuracy                           0.75       306
   macro avg       0.76      0.75      0.74       306
weighted avg       0.75      0.75      0.74       306
```

Figure 10:Classification Report for Efficient Net B2 model

The classification report includes accuracy, recall, and F1-scores for each class, which indicate the model's performance. The overall accuracy is 75%, and the macro-average F1-score of 0.74 indicates modest efficacy. Class 1 has the greatest performance (F1-score of 0.99), while class 6 has the lowest (0.54), showing misclassification concerns. The recall imbalance for specific classes, such as class 3 (0.65) and class 4 (0.39), indicates that the model has difficulty properly identifying certain categories. Improving feature representation, addressing class imbalance, or fine-tuning hyperparameters might improve overall performance.

The paper compares CNN, Xception, and EfficientNet-B2 models for smart contract vulnerability identification, using transfer learning to improve accuracy. CNN produces the best accuracy (71%), with steady training performance, whereas Xception, while effective (69%), has overfitting hazards. The EfficientNet-B2 model, a user-friendly online application, has considerably enhanced Ethereum security research by accurately categorizing the majority of instances in a dataset at 75%. However, class-specific measures are required to determine its benefits and drawbacks. This study adds to the creation of a more accessible tool for real-world use.

4.3. Comparative Analysis of all 4 models

The comparison table summarizes the performance of three different Deep Learning models: CNN, XCEPTION, and EfficientNet B2. While CNN achieves a moderate accuracy of 0.71, XCEPTION exhibits higher precision at 0.840, albeit with a slightly lower accuracy of 0.69. Notably, EfficientNet B2 outperforms both models with the highest accuracy of 0.75, along with balanced precision, recall, and F1-score values around 0.74. These results suggest that EfficientNet B2 may offer superior performance in classification tasks compared to the other models, emphasizing the importance of selecting an appropriate model architecture for specific application requirements.

Table 1:Performance Comparison of DL Models

Here is a table comparing the **weighted average** precision, recall, and F1-score for the three models:

Metric	CNN (71% Accuracy)	Xception (69% Accuracy)	EfficientNet-B2 (75% Accuracy)
Precision	0.73	0.70	0.75
Recall	0.71	0.69	0.75
F1-Score	0.71	0.69	0.74

EfficientNet-B2 outperforms all weighted average measures, giving it the top-performing model overall. CNN performs somewhat better than Xception but falls short of EfficientNet-B2.

5. CONCLUSION

The major objective of the research is classification of smart contract vulnerabilities with the help of a large dataset of about 2,000 Ethereum smart contracts from both the Ethereum documentation and the Smart Contract Dataset repository from GitHub. The methodology followed three primary modules. Module 1 was performed in Visual Studio Code where bytecode is extracted from Solidity files to produce images for deep learning model training. Module 2, which was done using Colab, concerns importing and preparing the image dataset and balancing class data through SMOTE. It included developing and testing three deep learning models: CNN; XCEPTION; and EfficientNet-B2. This Module 3 also produced a web application based on Flask for prediction of vulnerabilities, bytecode extraction, and user interaction. The Convolutional Neural Network (CNN) has made it possible to attain an overall accuracy of 71% in identifying smart contract vulnerabilities in a most reasonable model, where the sensitivity and specificity differed according to vulnerability classes. While XCEPTION performs with an overall accuracy of 69%, the model excels in recognizing samples from Class 1, EfficientNet-B2 is the one that boasts the highest accuracy of 75% thus establishing itself as the best in classifying smart contract vulnerabilities. The developed Flask web application is a user-friendly approach through which bytecode can be extracted and also vulnerabilities can be predicted from Solidity smart contract files.

REFERENCES

[1] Abdelaziz, T., Hobor, A., 2023. Smart Learning to Find Dumb Contracts (Extended Version). https://doi.org/10.48550/ARXIV.2304.10726

[2] Cai, J., Li, B., Zhang, J., Sun, X., Chen, B., 2022. Combine Sliced Joint Graph with Graph Neural Networks for Smart Contract Vulnerability Detection. SSRN Journal. https://doi.org/10.2139/ssrn.4074767

[3] Casucci, A., Mazzitelli, C., Tsiplakis, V., D'Arienzo, L., Breschi, L., Ferrari, M., 2023. Digital Impressions in Edentulous Patients: A Systematic Review for Clinical Evidence. Int J Prosthodont 36, 486–497. https://doi.org/10.11607/ijp.7483

[4] Chen, J.V., Chotimapruek, W., Ha, Q.-A., Widjaja, A.E., 2021. Investigating Female Customer's Impulse Buying in Facebook B2C Social Commerce: An Experimental Study. Contemporary Management Research 17, 65–96. https://doi.org/10.7903/cmr.20448

[5] Clack, C.D., Bakshi, V.A., Braine, L., 2016. Smart Contract Templates: foundations, design landscape and research directions. https://doi.org/10.48550/ARXIV.1608.00771

[6] Eshghie, M., Artho, C., Gurov, D., 2021. Dynamic Vulnerability Detection on Smart Contracts Using Machine Learning, in: Evaluation and Assessment in Software Engineering. Presented at the EASE 2021: Evaluation and Assessment in Software Engineering, ACM, Trondheim Norway, pp. 305–312. https://doi.org/10.1145/3463274.3463348

[7] Fadele Ayotunde Alaba, Hakeem Adewale Sulaimon, Madu Ifeyinwa Marisa, Owamoyo Najeem, 2023. Smart Contracts Security Application and Challenges: A Review. Cloud Computing and Data Science 15–41. https://doi.org/10.37256/ccds.5120233271

[8] Gohil, M.R., Maduskar, S.S., Gajria, V., Mangrulkar, R., 2021. Blockchain and Its Applications in Healthcare:, in: Ben Mnaouer, A., Fourati, L.C. (Eds.), Advances in Information Security, Privacy, and Ethics. IGI Global, pp. 271–294. https://doi.org/10.4018/978-1-7998-5839-3.ch012

[9] He, D., Deng, Z., Zhang, Y., Chan, S., Cheng, Y., Guizani, N., 2020. Smart Contract Vulnerability Analysis and Security Audit. IEEE Network 34, 276–282. https://doi.org/10.1109/MNET.001.1900656

[10] Jiang, B., Liu, Y., Chan, W.K., 2018. ContractFuzzer: fuzzing smart contracts for vulnerability detection, in: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. Presented at the ASE '18: 33rd ACM/IEEE International Conference on Automated Software Engineering, ACM, Montpellier France, pp. 259–269. https://doi.org/10.1145/3238147.3238177

[11] Khan, S.N., Loukil, F., Ghedira-Guegan, C., Benkhelifa, E., Bani-Hani, A., 2021. Blockchain smart contracts: Applications, challenges, and future trends. Peer-to-Peer Netw. Appl. 14, 2901–2925. https://doi.org/10.1007/s12083-021-01127-0

[12] Liao, J.-W., Tsai, T.-T., He, C.-K., Tien, C.-W., 2019. SoliAudit: Smart Contract Vulnerability Assessment Based on Machine Learning and Fuzz Testing, in: 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS). Presented at the 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS), IEEE, Granada, Spain, pp. 458–465. https://doi.org/10.1109/IOTSMS48152.2019.8939256

[13] Liu, H., Mohd, Yasin, M., Ruan, Q., 2023. A study on how social media influences on impulsive buying. Expert Systems. https://doi.org/10.1111/exsy.13448

[14] Macrinici, D., Cartofeanu, C., Gao, S., 2018. Smart contract applications within blockchain technology: A systematic mapping study. Telematics and Informatics 35, 2337–2354. https://doi.org/10.1016/j.tele.2018.10.004

[15] Mezina, A., Ometov, A., 2023. Detecting Smart Contract Vulnerabilities with Combined Binary and Multiclass Classification. Cryptography 7, 34. https://doi.org/10.3390/cryptography7030034

[16] Sharad Mangrulkar, R., Vijay Chavan, P., 2024. Ethereum Blockchain, in: Blockchain Essentials. Apress, Berkeley, CA, pp. 123–166. https://doi.org/10.1007/978-1-4842-9975-3_4

[17] Sun, S., Cao, Z., Zhu, H., Zhao, J., 2020. A Survey of Optimization Methods From a Machine Learning Perspective. IEEE Trans. Cybern. 50, 3668–3681. https://doi.org/10.1109/TCYB.2019.2950779

[18] Turakhia, A., Date, C., Correia, C., Mangrulkar, R., Williams, I., Mahalle, P., 2023. Improving Product Traceability and Security in Supply Chain Management using BlockChain, in: 2023 International Conference on Advanced Computing Technologies and Applications (ICACTA). Presented at the 2023 International Conference on Advanced Computing Technologies and Applications (ICACTA), IEEE, Mumbai, India, pp. 1–6. https://doi.org/10.1109/ICACTA58201.2023.10393309

[19] Wohrer, M., Zdun, U., 2018. Smart contracts: security patterns in the ethereum ecosystem and solidity, in: 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE). Presented at the 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE), IEEE, Campobasso, pp. 2–8. https://doi.org/10.1109/IWBOSE.2018.8327565

[20] Zhuang, Y., Liu, Z., Qian, P., Liu, Q., Wang, X., He, Q., 2020. Smart Contract Vulnerability Detection using Graph Neural Network, in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. Presented at the Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence {IJCAI-PRICAI-20}, International Joint Conferences on Artificial Intelligence Organization, Yokohama, Japan, pp. 3283–3290. https://doi.org/10.24963/ijcai.2020/454