INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

Securing Smart Contracts: Harnessing the Power of Efficient NetB2 Detection

Janhavi Satam[1]*, **Sangeeta Vhatkar**[1]

1 Department of Information Technology, Thakur College of Engineering and Technology, Mumbai, 400101, Maharashtra, India

OPEN ACCESS

Received: 01-03-2024

Accepted: 21-03-2024

Published: 15-04-2024

Citation: Satam J, Vhatkar S (2024) Securing Smart Contracts: Harnessing the Power of Efficient NetB2 Detection. Indian Journal of Science and Technology 17(15): 1621-1632. https://doi.org/10.17485/IJST/v17i15.617

* **Corresponding author**.

janhavisatam1999@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objective: Using a variety of datasets from the Ethereum documentation and Smart Contract Dataset repository, this study tackles the crucial problem of classifying smart contract vulnerabilities. **Methods:** Our study uses a three-module method and focuses on the Resource 3 Dataset, which contains over 2,000 Ethereum smart contracts, including inherited contracts. The groundwork for deep learning model training is laid in Module 1 by extracting bytecode from Solidity files and creating images thereafter. In Colab, Module 2 entails importing data, pre-processing, SMOTE balancing, and building three deep learning models: CNN, XCEPTION, and EfficientNet-B2. Module 3 is a Flask-based web application created in Visual Studio Code that enables vulnerability predictions, bytecode extraction, and user interaction. **Findings:** With an overall accuracy of 71 percent, the Convolutional Neural Network (CNN) displays its effectiveness in classifying vulnerabilities. Although the accuracy of XCEPTION and EfficientNet-B2 is 69% and 75%, respectively, the latter is the top performer. **Novelty & Applications:** The online application adds to the comprehensive examination of smart contract security by giving users an easy-to-use interface. The EfficientNet-B2 model stands out as a dependable tool for precise vulnerability classification, and this study advances our understanding of and efforts to mitigate vulnerabilities in Ethereum smart contracts.

Keywords: Smart Contracts; Vulnerability Classification; Ethereum; Deep Learning; Convolutional Neural Network (CNN)

1 Introduction

The introduction sets the stage by delineating the critical problem of smart contract vulnerabilities and the existing gap in the current research landscape. Despite the growing prominence of smart contracts, their susceptibility to security threats poses a significant challenge. There are so many studies done on smart contract vulnerability but still, there are several issues which have not been resolved yet properly. The problem statement in smart contract vulnerability research revolves around the persistent threat to the security of Ethereum and similar blockchain ecosystems. The existing gap lies in the limitations of current methodologies, marked by insufficient accuracy and a lack of unified approaches for precise vulnerability classification like previous research[1] on smart contract vulnerabilities has seen a limited exploration of the

XCEPTION and EfficientNet-B2 models. Our proposed approach uniquely emphasizes these two advanced models, and both exhibit impressive accuracy. The underutilization of XCEPTION and EfficientNet-B2 in prior studies underscores the novelty of our research, as we delve into uncharted territory to harness their potential. This research introduces a unique approach by conducting feature extraction directly from the smart contract file. This distinctive methodology sets this work apart, as I focus on extracting crucial features directly from the contract file itself. Initially, this research extracts the bytecode directly from the solidity file of the smart contract. So, this work intends to address those gaps which have been not resolved earlier.

Despite their limited prior exposure, both models showcase notable performance in our study, highlighting their effectiveness in smart contract vulnerability classification. This distinctive focus on XCEPTION and EfficientNet-B2 adds a pioneering dimension to our research, advancing the understanding and applicability of these models in bolstering the security of Ethereum smart contracts. In addressing prevalent challenges within smart contract security, prior works have grappled with limited accuracy and comprehensive vulnerability classification. Existing studies lack a unified approach to leverage advanced models. In(1), proposed the Deep Learning Vulnerability Analyzer (DLVA) which demonstrates impressive performance in detecting vulnerabilities in Ethereum smart contracts, there are potential gaps and considerations in this research. Firstly, the study's focus on Ethereum smart contracts may limit its applicability to other blockchain platforms with different bytecode structures. Adapting DLVA to accommodate diverse blockchain ecosystems could enhance its versatility. Additionally, the evaluation of DLVA primarily against state-of-the-art alternatives may overlook certain aspects. The benchmarks may not cover all possible scenarios and may not reflect the dynamic nature of emerging vulnerabilities. So that is why our study will use the same Ethereum smart contracts on vulnerability detection using different deep learning models. So, Smart contracts are becoming an essential aspect of blockchain technology, especially in the Ethereum ecosystem(2). They are self-executing contracts with the conditions of the agreement explicitly built into the code. It is crucial to make sure these contracts are secure since they manage sensitive activities, decentralised apps, and financial transactions(3). Ethereum smart contract vulnerability classification has become an important field of study to locate and reduce possible hazards present in these electronic contracts. Smart contracts are vulnerable to many vulnerabilities due to the inherent complexities and problems introduced by the decentralised nature of blockchain technology(4). If these weaknesses are taken advantage of, there may be monetary losses, illegal access, and other security lapses. The necessity to categorise and evaluate these vulnerabilities methodically has prompted academics to investigate novel strategies that make use of deep learning models. The expanding use of smart contracts for a variety of purposes, from supply chain management to financial transactions, is the driving force behind this study(5). As blockchain technology becomes more widely used, there is a growing need to improve smart contract security. Because blockchain networks are decentralised and dynamic, traditional vulnerability assessment techniques might not be able to fully address their shortcomings. Using deep learning models offers a viable way to thoroughly examine the patterns and underlying complexity of smart contract code.

1.1 Analysing Smart Contract Measures

A smart contract can be automatically executed and that needs to be carried out by its specifications. While smart contracts can be carried out automatically by computers, some tasks still need human input and supervision(6). To solve the security issues related to smart contracts on the Ethereum blockchain,(7) suggested SoliAudit (Solidity Audit) in their study. Understanding that smart contracts are vulnerable to hacking—a fact made abundantly clear by the DAO attack in 2016—they used machine learning and fuzz testing to conduct a thorough vulnerability assessment. Using Solidity machine code as learning features, and machine learning technology, SoliAudit verified 13 different types of vulnerabilities that corresponded with the Top 10 dangers that an open security group had identified. Their technique was noteworthy for incorporating a grey-box fuzz testing tool for online transaction verification. This mechanism included fuzzier contracts and a simulated blockchain environment. In contrast to earlier systems, SoliAudit showed that it could find vulnerabilities without the need for specialised expertise or pre-established patterns. An astounding 90% accuracy rate was found in the study, which was carried out on around 18,000 smart contracts from the Ethereum blockchain and Capture-the-Flag samples. Moreover, the SoliAudit fuzzing feature demonstrated efficacy in pinpointing possible vulnerabilities, such as reentrancy and arithmetic overflow problems. (8) has tackled the growing difficulties brought on by the platform's greater scalability in its investigation of Ethereum's innovative smart contract technology, which has resulted in a boom of decentralised apps. Taking note of the increased susceptibility to security risks, their paper offers an extensive analysis of several weaknesses in Ethereum smart contracts as well as related mitigation strategies that are common in both scholarly and commercial research. The vulnerability concerning random numbers in contracts that mimic the Fomo3d game is examined in detail, along with the attack and defense strategies used. Security issues still exist despite smart contract technology's improvements, which is why(8) looked at the Ethereum smart contract security audit techniques that are now in use. They compare popular auditing tools to highlight the advantages and disadvantages of each strategy. The present research enhances the comprehension of Ethereum smart contract vulnerabilities by providing valuable perspectives on possible risks and efficacious countermeasures in the dynamic domain of decentralised apps. When it comes to smart contract vulnerabilities, especially those created for the Ethereum blockchain, [9] carried out an extensive study of 21,270 problematic contracts that were found by six recent research studies. Unlike what was expected given the number of contracts identified as susceptible, their results show that the actualization of these vulnerabilities is rather low, with only 504 out of 21,270 contracts being exploited. This amounts to a maximum of 9,066 ETH (or around 1.8 million USD), which is about 0.29 per cent of the 3 million ETH (or 600 million USD) that some of the papers under review claimed. Although [9] acknowledges the need to research smart contract vulnerabilities, it speculates that the possible consequences of faulty programming may have been exaggerated.

1.2 Enhancing Smart Contract Vulnerability Detection

To tackle the widespread security issues surrounding smart contracts and the significant financial damages caused by flaws, [10] put forth a unique method that makes use of graph neural networks (GNNs) to improve detection precision. As part of their process, they build a contract graph that captures the syntactic and semantic structures of a smart contract function. The network is then normalised and critical nodes are highlighted through an elimination phase. To enable learning from the normalised graphs for efficient vulnerability identification, [10] presented a temporal message propagation network (TMP) and a degree-free graph convolutional neural network (DR-GCN). As worries about smart contract vulnerabilities have grown, as have the significant monetary losses brought about by hacker assaults, [11] put forth a novel solution that enhances smart contract vulnerability detection by fusing expert knowledge with graph neural networks (GNNs). Taking into consideration the drawbacks of previous security analysis attempts that relied on laborious and unscalable expert-defined rules, sought to close the gap by utilising the rich control- and data-flow semantics of source code by constructing an extensive contract graph. To normalise the graph and highlight important nodes, the suggested technique first included a node deletion phase. Next, a new temporal message propagation network was included to extract graph properties. Interestingly, the method combined expert-defined security patterns with deep neural networks, offering a comprehensive approach to accurate and scalable vulnerability identification. Empirical assessments carried out on smart contracts from the Ethereum and VNT Chain platforms revealed notable advancements in accuracy compared to cutting-edge techniques, with detection accuracy for reentrancy, timestamp dependence, and infinite loop vulnerabilities reaching 89.15 per cent, 89.02 per cent, and 83.21 per cent, respectively [11].

1.3 Advancements in Smart Contract Vulnerability Detection

Because smart contracts can result in significant financial losses, the use of smart contracts in blockchain technology has increased recently, which has increased the attention on smart contract vulnerability detection. Traditional approaches relying on fixed expert-defined criteria face limitations in terms of generality, scalability, and accuracy. [12] conducted an empirical study on ChatGPT's performance in identifying smart contract vulnerabilities, revealing high recall rates but limited precision. [13] proposed a mechanism integrating graph neural networks and expert patterns to enhance smart contract vulnerability detection, showcasing superiority over existing methods. Meanwhile, [14] introduced ASSBert, a framework combining active and semi-supervised learning for efficient vulnerability detection with limited labelled data. [15] presented a white box knowledge-enforcing methodology, outperforming existing schemes in multimodal vulnerability mining. [16] proposed a Graph Neural Network-based approach, achieving an impressive 89.2% precision and 92.9% recall in smart contract vulnerability detection.

1.4 Ethereum Blockchain in Smart Contract Vulnerability Detection

Blockchain is an underlying type of technology which is having different types of cryptocurrencies including Ethereum and Bitcoin which serve as the foundation of smart contracts. There is an open, decentralized source platform called Ethereum which enables the execution and creation of smart contracts and it was introduced by Vitalik Buterin in 2015 [17]. In the commercial markets, it gives new values to the word which is 'trust' and it also comprises shared different blocks of transactions and extremely secure [18].

2 Methodology

We provide a multi-step methodology for classifying smart contract vulnerabilities that capitalises on deep learning models' advantages. First, we take bytecode out of Solidity files for smart contracts, which is a basic representation of the contract. After that, we use this bytecode to create pictures, which serve as the foundation for our deep learning models' training. By doing this, we hope to extract complex patterns and characteristics from the bytecode that will help the models identify vulnerabilities more accurately. The implementation of deep learning models, namely Convolutional Neural Network (CNN), Xception, and EfficientNet-B2, is the next stage. By employing the produced pictures for training, these models can comprehend and grasp the correlations found in the data. The models provide a reliable method for classifying smart contract vulnerabilities once they have been trained and used for predictions. The model-derived predictions offer valuable insights into the possible weaknesses found in the smart contracts under analysis. This combined method, which uses deep learning models, picture synthesis, and bytecode extraction, aims to improve the precision and effectiveness of smart contract vulnerability classification. The experimental data and analyses will be covered in detail in the next sections of this paper, giving readers a thorough knowledge of how successful our suggested technique is. I also split our data into 90:10 ratios for training and testing but no validation was used(19) (20).

2.1 Module 1: Bytecode Extraction and Image Generation

Module 1's main objective is to convert Solidity files for Ethereum smart contracts into a format that can be used to train deep learning models. This module consists of taking the bytecode out of the Solidity files and turning it into pictures. Deep learning models will be trained using the produced photos as input data.

1. **Bytecode Extraction:** I programmatically extract bytecode from Ethereum smart contract Solidity files using the Solcx library. The expedited process of compiling Solidity source code and retrieving the related bytecode is made possible by Solcx. The low-level instructions of the smart contract are represented by this bytecode.
2. **Image Generation through Bytecode:** After that, the extracted bytecode is processed to produce pictures. The bytecode translates each opcode or group of opcodes into a distinct visual representation. Through this conversion process, I can extract complex patterns and connections from the bytecode and transform them into a format that can be used to train deep learning models that are based on images.

2.2 Module 2: Data Preprocessing and Model Development (COLAB)

1. **Importing Libraries and Data Loading:** I start by importing the necessary libraries, such as scikit-learn for preprocessing and evaluation tasks, Pandas and Numpy for data manipulation, and OpenCV for loading picture datasets. Using OpenCV, the picture dataset is imported, and rudimentary exploratory data analysis is carried out to comprehend the distribution of classes and pinpoint any difficulties.
2. **Data Preprocessing:** Data labelling, label binarization, and picture scaling are examples of preprocessing procedures. We utilise the Synthetic Minority Over-sampling Technique (SMOTE) to rectify the imbalance in classes, guaranteeing a more equitable portrayal of susceptible classes.
3. **Dataset Description:** The dataset used in this research is an extensive set of Ethereum smart contracts—more than 2,000 examples total—including inherited contracts. The dataset was carefully cleaned before being narrowed down to seven different categories of vulnerabilities that are frequently present in smart contracts. This collection provides many examples of actual situations and security issues that may arise on the Ethereum network. The dataset offers a wealth of information for training and assessing our smart contract vulnerability classification models. It is sourced from the Ethereum documentation and the Smart Contract Dataset repository on GitHub. It contains detailed information on the bytecode of the contracts, which is essential to our image-based method.
4. **Model Development:** Three deep learning models are put into practice: EfficientNet-B2, Xception, and Convolutional Neural Network (CNN). These models were chosen because they have a track record of success in picture classification tasks. TensorFlow and Keras are leveraged in the training process.
5. **Model Evaluation:** Using common assessment criteria including confusion matrices, classification reports, and specificity & and sensitivity studies, the effectiveness of each model is evaluated. These metrics shed light on how well the models categorise various vulnerability classifiers.

A countplot of distribution of target classes before the use of SMOTE is shown in Figure 1. (Synthetic Minority Over-sampling Technique). The x-axis displays the many target classes, which include reentrancy (RE), stringent equality (SE), block number dependency (BN), timestamp dependency (TP), and other comparable categories. The y-axis, which runs from 0 to 400, shows the number of occurrences within each class. It is clear from watching the plot that there is a clear class divide, with certain categories having far fewer instances than others. Machine learning models may perform worse as a result of this imbalance since they may find it difficult to forecast underrepresented classes. This work uses SMOTE methods, as shown in Figure 2, to artificially balance the distribution of classes in order to overcome this problem.

A countplot depicting the intended class distribution is shown in Figure 2 following the use of SMOTE (Synthetic Minority Over-sampling Technique) for data balancing. By contrasting Figure 1 (pre-SMOTE) and 2 (post-SMOTE), it is clear that the class disparity has been lessened. By effectively increasing the number of minority classes, SMOTE has made sure that all categories are represented more fairly.

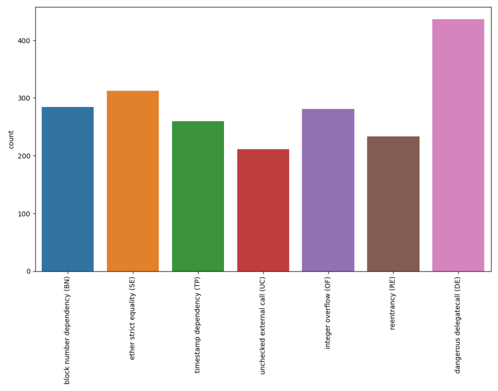


Fig 1. Countplot of target class before applying SMOTE

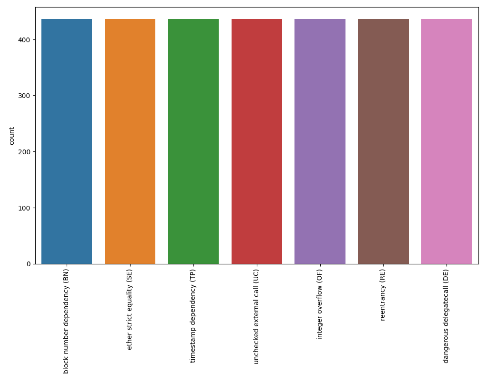
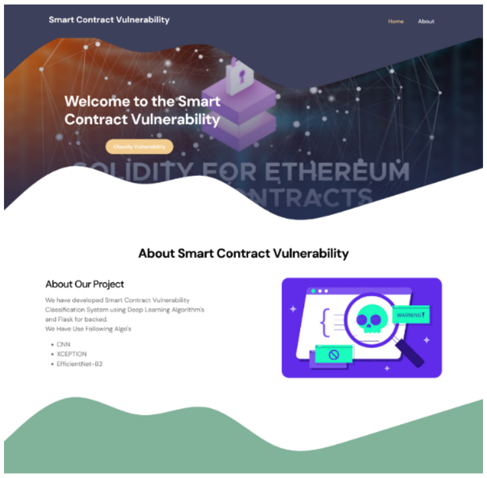


Fig 2. Countplot of target class after applying SMOTE

2.3 Module 3: Web Application Development (VS CODE)

We utilise Flask to construct a web application that improves user engagement. With this application, users may enter smart contract files, extract bytecode, and obtain predictions from our trained models on vulnerability classification. The user experience is made smooth and responsive with the Flask framework. Figure 3 is the home page for the web application of smart contract vulnerability.


Smart Contract Vulnerability
Home
About
Welcome to the Smart Contract Vulnerability
Classify Vulnerability
SOLIDITY FOR ETHEREUM
CONTRACTS
WARNING!

About Smart Contract Vulnerability

About Our Project

We have developed Smart Contract Vulnerability Classification System using Deep Learning Algorithm's and Flask for backed.
We Have Use Following Algo's

- CNN
- XCEPTION
- EfficientNet-B2

Fig 3. Web application home page

Figure 4 shows selecting the solidity file and then clicking on the predicted sol file which is been uploaded.

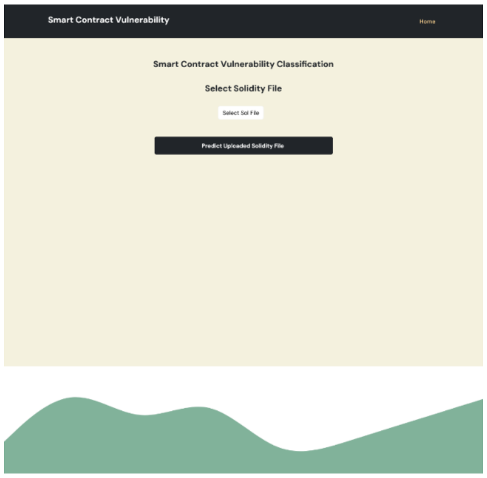
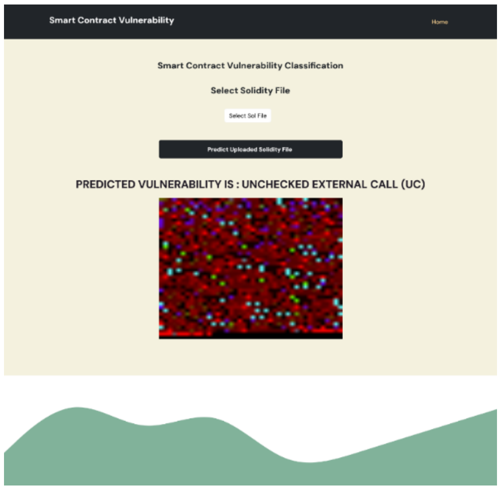

Smart Contract Vulnerability
Home
Smart Contract Vulnerability Classification
Select Solidity File
Select Sol File
Predict Uploaded Solidity File

Fig 4. Selecting the Sol file to predict the uploaded Sol file

Figure 5 predicts the vulnerability which is unchecked external call (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)

Fig 5. Predicting Vulnerability is: Unchecked External Call (UC)

3 Results and Discussion

3.1 CNN

With an overall accuracy of 71 per cent, the Convolutional Neural Network (CNN) demonstrated its capacity to categorise most of the dataset's cases accurately. This accuracy points to a respectable degree of performance in differentiating between various groups within the data. But accuracy by itself doesn't give a whole picture of the model's performance; to grasp the model's efficacy across other categories, one must go further into class-specific measures. In terms of class sensitivity, CNN showed different degrees of proficiency in accurately recognising examples of every class. Of particular note, class 2 had the maximum sensitivity of 99.62 per cent, suggesting that the model is very capable of correctly identifying occurrences that fall into this group. Classes 0 and 1 further had comparatively high sensitivity values of 96.18 per cent and 98.09 per cent, respectively, indicating competent performance in identifying cases from these classes. Classes 3, 4, 5, and 6 presented the greatest challenge for the model, with sensitivity levels ranging from 87.02 per cent to 94.68 per cent. This points to possible areas for development, particularly about correctly detecting instances from these classes. In terms of specificity, the CNN showed different levels of proficiency in accurately detecting negative cases for every class. With specificity levels of 97.73 per cent and 84.09 per cent, respectively, classes 0 and 1 stood out as having the highest capacity to effectively filter out occurrences that did not fall into these classes. However, the specificity levels for classes 2, 5, and 6 were lower, ranging from 72.09 per cent to 79.55 per cent, indicating that these categories may have difficulties differentiating between negative examples. With a specificity of only 34.09 per cent, Class 4 had the lowest performance, suggesting that there is still much space for development in terms of correctly detecting negative cases for this class.

The CNN Accuracy Graph, Figure 6, shows how the model performed over epochs for both training and validation. The training accuracy is shown by the blue line, while the validation accuracy is indicated by the red line. The training accuracy often rises with the length of each epoch, suggesting that the model is picking up new information from the training set. The red-coloured validation accuracy indicates how effectively the model generalises to previously unknown data. Both lines should ideally show an increasing trend, which denotes learning that is efficient without being overfitting.

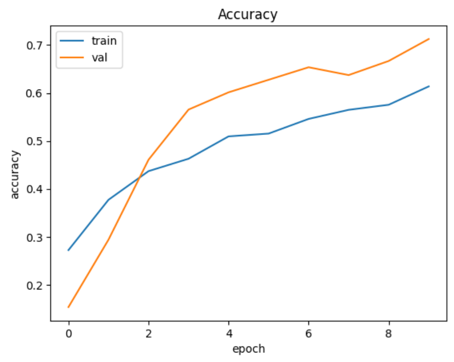


Fig 6. Accuracy Graph

Figure 7, the CNN Loss Graph, shows the loss during epochs during training and validation. The training loss is shown by the blue line, while the validation loss is shown by the red line. An improvement in model learning is shown by a decrease in both training and validation loss.

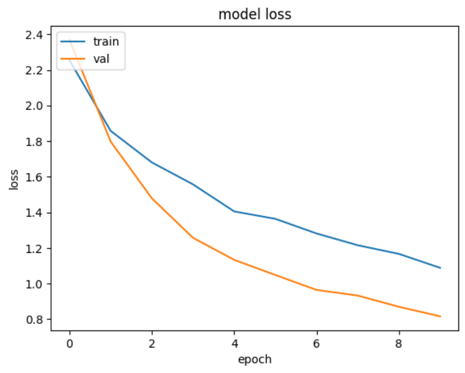
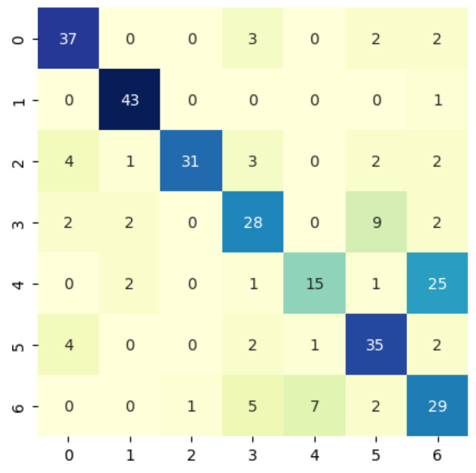


Fig 7. Loss Graph

A CNN model's confusion matrix is shown in Figure 8. The true class is represented by each row, while the predicted class is represented by each column. Correctly classified cases are represented by diagonal elements, whereas incorrectly classified examples are shown by off-diagonal elements. The model's performance is clearly conveyed via the heat map with annotations, which highlights regions of correct predictions and possible misunderstanding.

3.2 XCEPTION

With an overall accuracy of 69 per cent, the XCEPTION model demonstrated its ability to categorise most of the dataset's cases accurately. Despite being marginally less accurate than CNN, this still points to a respectable degree of competence in



	0	1	2	3	4	5	6
0	37	0	0	3	0	2	2
1	0	43	0	0	0	0	1
2	4	1	31	3	0	2	2
3	2	2	0	28	0	9	2
4	0	2	0	1	15	1	25
5	4	0	0	2	1	35	2
6	0	0	1	5	7	2	29

Fig 8. Confusion Matrix

differentiating across classes. To acquire a more thorough grasp of the model's efficacy across several categories, it is imperative to dig further into class-specific indicators, as with any study. When class sensitivity was investigated, the XCEPTION model showed different degrees of success in accurately classifying occurrences. Interestingly, class 1 had the highest sensitivity of any category, at 99.62 per cent, suggesting that the model could recognise occurrences from this group with remarkable accuracy.

In Figure 9, the Accuracy Graph of XCEPTION, this graph illustrates the model's learning progress, aiming for higher accuracy. The values range from 0 to 0.8, indicating the proportion of correctly predicted instances. A rising trend in both curves signifies effective training without overfitting.

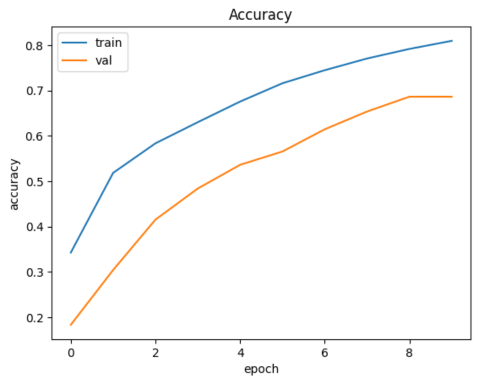


Fig 9. Accuracy Graph

Simultaneously, Figure 7, the Loss Graph for XCEPTION, the loss values range from 0 to 2.0, reflecting the discrepancy between predicted and actual values.

Figure 11 presents a Confusion Matrix for the XCEPTION model. The matrix is displayed as a heat map, with rows and column. The annotations provide a quantitative measure of model behavior, aiding in the identification of specific areas where the XCEPTION model excels or may require improvement in classification tasks.

3.3 EFFICIENT NET-B2

With the greatest accuracy of 75% among the models tested, the EfficientNet-B2 model stands out as the best performer. This remarkable accuracy highlights how well the model classified most of the cases in the dataset correctly, demonstrating its better performance over competing models. Accuracy, however, only gives a broad picture of the model's performance; to learn more about the model's advantages and disadvantages, class-specific metrics must be examined.

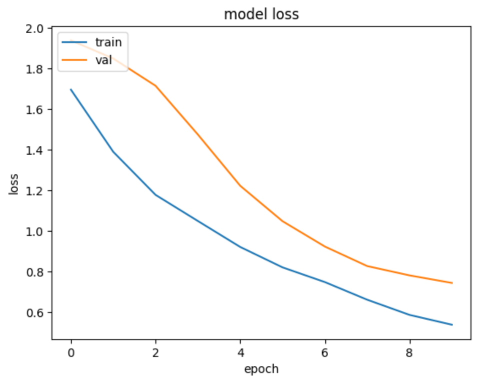


Fig 10. Loss Graph

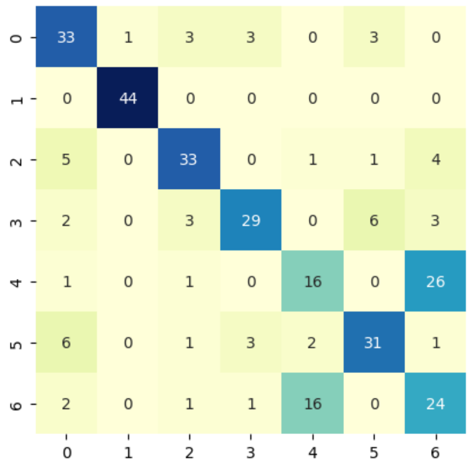


Fig 11. Confusion Matrix

In Figure 12, the Accuracy Graph of EFFICIENT NET B2, the upward trend in both curves suggests the model is learning effectively, aiming for higher accuracy without significant overfitting.

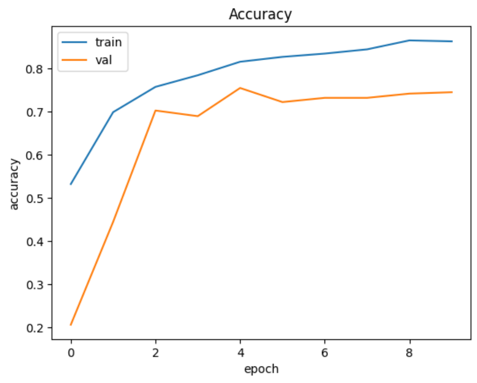


Fig 12. Accuracy Graph

Simultaneously, in Figure 7, the Loss Graph for EFFICIENT NET B2, a declining trend in both curves indicates effective learning, while the convergence between training and validation losses reflects good generalization.

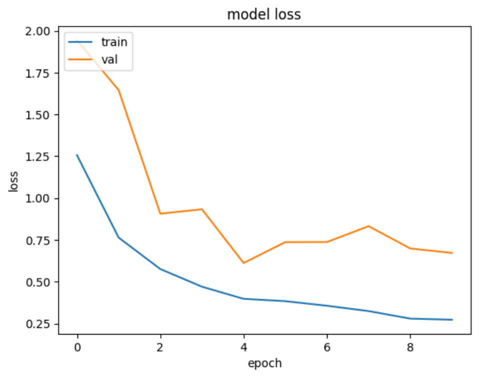
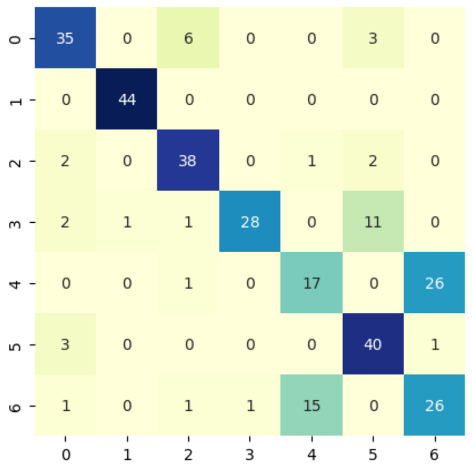


Fig 13. Loss Graph

Figure 14 displays the Confusion Matrix for the EFFICIENT NET B2 model. Visualized as a heatmap, rows represent true classes, columns indicate predicted classes, and each cell is annotated with the count of instances. The color intensity, ranging from light to dark blue, signifies the magnitude of correct and incorrect predictions.



	0	1	2	3	4	5	6
0	35	0	6	0	0	3	0
1	0	44	0	0	0	0	0
2	2	0	38	0	1	2	0
3	2	1	1	28	0	11	0
4	0	0	1	0	17	0	26
5	3	0	0	0	0	40	1
6	1	0	1	1	15	0	26

Fig 14. Confusion Matrix

This work's novelty lies in its three-module approach, employing CNN, XCEPTION, and the superior EfficientNet-B2 model to classify Ethereum smart contract vulnerabilities. The study not only achieves a commendable overall accuracy of 75% but also delves into class-specific metrics, revealing EfficientNet-B2's exceptional performance. The creation of a user-friendly web application further enhances accessibility, contributing a valuable tool for real-world applications. This research advances the field by providing a comprehensive solution to the critical problem of smart contract vulnerability classification, with EfficientNet-B2 emerging as a standout model for precise and reliable assessments, marking a significant stride in Ethereum security research.

3.4 Comparative Analysis of all 4 models

The comparison Table 1 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

Model	Accuracy	Precision	Recall	F1-score
CNN	0.71	0.733	0.756	0.745
XCEPTION	0.69	0.840	0.729	0.781
EfficientNet B2	0.75	0.750	0.740	0.740

4 Conclusion

Using an extensive dataset taken from the Ethereum documentation and the Smart Contract Dataset repository on GitHub, we tackled the important job of smart contract vulnerability classification in this work. Our investigation included approximately 2,000 Ethereum smart contracts, including inherited contracts and post-cleaning operations, with a focus on the Resource 3 Dataset in the repository. Three main modules comprised our methodology: In Module 1, which was run in Visual Studio Code, bytecode was extracted from Solidity files and then pictures were generated to train deep learning models. The second module, which was executed in Colab, involved importing and preparing the picture dataset, utilising SMOTE to balance class data, and building and assessing three deep learning models: CNN, XCEPTION, and EfficientNet-B2. Ultimately, Module 3, created using Flask and VS Code, produced a web application that enabled vulnerability predictions, bytecode extraction, and user interaction. With an overall accuracy of 71 per cent, the Convolutional Neural Network (CNN) proved its efficacy in classifying smart contract vulnerabilities. CNN proved to be a reliable model, despite differences in sensitivity and specificity among various vulnerability classes. Even though XCEEPTION accuracy was just 69%, it still performed admirably, especially when it came to identifying examples from class 1. But with the greatest accuracy of 75%, the EfficientNet-B2 model outperformed the rest, making it the top performer in the categorization of smart contract vulnerabilities. The Flask-powered web application functions as an intuitive user interface for bytecode extraction and vulnerability prediction from Solidity files used in smart contracts.

References

1) Abdelaziz T, Hobor A. Smart Learning to Find Dumb Contracts. 2023. Available from: https://doi.org/10.48550/arXiv.2304.10726.
2) Wohrer M, Zdun U. Smart contracts: security patterns in the ethereum ecosystem and solidity. In: 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE);vol. 2018. IEEE. 2018;p. 2–8. Available from: https://ieeexplore.ieee.org/document/8327565.
3) Garg R. Ethereum based Smart Contracts for Trade and Finance. In: International Conference on Blockchain, Smart Contracts and Cryptocurrencies (ICBSC 2022);vol. 16. Zenodo. 2022;p. 1–19. Available from: https://zenodo.org/doi/10.5281/zenodo.5854729.
4) Alaba FA, Sulaimon HA, Marisa MI, Najeem O. Smart Contracts Security Application and Challenges: A Review. *Cloud Computing and Data Science*. 2023;5(1):15–41. Available from: https://dx.doi.org/10.37256/ccds.5120233271.
5) Khan SN, Loukil F, Ghedira-Guegan C, Benkhelifa E, Bani-Hani A. Blockchain smart contracts: Applications, challenges, and future trends. *Peer-to-Peer Networking and Applications*. 2021;14(5):2901–2925. Available from: https://dx.doi.org/10.1007/s12083-021-01127-0.
6) Turakhia A, Date C, Correia C, Mangrulkar R, Williams I, Mahalle P. Improving Product Traceability and Security in Supply Chain Management using BlockChain. In: 2023 International Conference on Advanced Computing Technologies and Applications (ICACTA). IEEE. 2024;p. 1–6. Available from: https://doi.org/10.1109/ICACTA58201.2023.10393309.
7) Liao JW, Tsai TT, He CK, Tien CW. 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). IEEE. 2019;p. 458–465. Available from: https://doi.org/10.1109/IOTSMS48152.2019.8939256.
8) He D, Deng Z, Zhang Y, Chan S, Cheng Y, Guizani N. Smart Contract Vulnerability Analysis and Security Audit. *IEEE Network*. 2020;34(5):276–282. Available from: https://dx.doi.org/10.1109/mnet.001.1900656.
9) Zhuang Y, Liu Z, Qian P, Liu Q, Wang X, He Q. Smart Contract Vulnerability Detection using Graph Neural Network. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. International Joint Conferences on Artificial Intelligence Organization. 2020;p. 3283–3290. Available from: https://doi.org/10.24963/ijcai.2020/454.
10) Liu Z, Qian P, Wang X, Zhuang Y, Qiu L, Wang X. Combining Graph Neural Networks with Expert Knowledge for Smart Contract Vulnerability Detection. *IEEE Transactions on Knowledge and Data Engineering*. 2023;35(2):1296 –1310. Available from: https://dx.doi.org/10.1109/tkde.2021.3095196.
11) Jiang B, Liu Y, Chan WK. ContractFuzzer: fuzzing smart contracts for vulnerability detection. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. ACM. 2018;p. 259–269. Available from: https://doi.org/10.1145/3238147.3238177.
12) Chen C, Su J, Chen J, Wang Y, Bi T, Wang Y, et al. When ChatGPT Meets Smart Contract Vulnerability Detection: How Far Are We?. 2023. Available from: https://doi.org/10.48550/arXiv.2309.05520.
13) Liu Z, Jiang M, Zhang S, Zhang J, Liu Y. A Smart Contract Vulnerability Detection Mechanism Based on Deep Learning and Expert Rules. *IEEE Access*. 2023;11:77990–77999. Available from: https://dx.doi.org/10.1109/access.2023.3298048.
14) Sun X, Tu L, Zhang J, Cai J, Li B, Wang Y. ASSBert: Active and semi-supervised bert for smart contract vulnerability detection. *Journal of Information Security and Applications*. 2023;73:103423. Available from: https://dx.doi.org/10.1016/j.jisa.2023.103423.
15) Jie W, Chen Q, Wang J, Koe ASV, Li J, Huang P, et al. A novel extended multimodal AI framework towards vulnerability detection in smart contracts. *Information Sciences*. 2023;636:118907. Available from: https://dx.doi.org/10.1016/j.ins.2023.03.132.
16) Cai J, Li B, Zhang J, Sun X, Chen B. Combine sliced joint graph with graph neural networks for smart contract vulnerability detection. *Journal of Systems and Software*. 2023;195:111550. Available from: https://dx.doi.org/10.1016/j.jss.2022.111550.

17) Mangrulkar RS, Chavan PV. Ethereum Blockchain. In: Blockchain Essentials. Berkeley, CA, USA. Apress. 2024;p. 123–166. Available from: https://doi.org/10.1007/978-1-4842-9975-3_4.

18) Gohil MR, Maduskar SS, Gajria V, Mangrulkar R. Blockchain and Its Applications in Healthcare. In: Enabling Blockchain Technology for Secure Networking and Communications. IGI Global. 2021;p. 271–294. Available from: https://doi.org/10.4018/978-1-7998-5839-3.ch012.

19) Eshghie M, Artho C, Gurov D. Dynamic Vulnerability Detection on Smart Contracts Using Machine Learning. In: Proceedings of the 25th International Conference on Evaluation and Assessment in Software Engineering. New York, NY, United States. ACM. 2021;p. 305–312. Available from: https://doi.org/10.1145/3463274.3463348.

20) Mezina A, Ometov A. Detecting Smart Contract Vulnerabilities with Combined Binary and Multiclass Classification. *Cryptography*. 2023;7(3):1–20. Available from: https://dx.doi.org/10.3390/cryptography7030034.