An Evaluation of Lightweight CNNs for Smart Contract Vulnerability Detection

***Abstract*—The proposed work investigates the use of lightweight convolutional neural networks (CNNs) for detecting vulnerability patterns in Solidity RGB-encoded smart contracts. Unlike heavy CNN models, which can be computationally intensive and fall short of optimal accuracy levels, the proposed study emphasizes efficiency. Transforming smart contract source code into RGB images not only reinforces security and protects proprietary information but also addresses compactness concerns, enabling convenient storage on online platforms. This approach ensures efficient use of bandwidth, enabling rapid scanning of contracts for potential vulnerabilities post deployment. The streamlined mechanism allows for quick and simultaneous assessment of thousands of contracts within seconds, a task that proves challenging with rigorous formal verification tools. This methodology aligns with the need for both security and efficiency in the dynamic landscape of smart contract development and deployment.**

***Index Terms*—Blockchain, Smart Contracts, Vulnerability Scanning, Convolution Neural Networks (CNNs)**

I. Introduction

RCENT years have witnessed a wave of interest in cryptocurrencies prompted by the extraordinary success of Bitcoin, which gave rise to a variety of cryptocurrency platforms, including Ethereum and Hyperledger [1]. Smart contracts are automated and decentralized programs that establish terms and rules for transactions involving buyers and sellers or producers and consumers. Smart contracts present specific challenges in spite of their many benefits, which include cost-effectiveness, security, accuracy, efficiency, trust, and transparency [2]. The severity of challenges arise from error-prone development life cycles and constraints of programming languages such as Solidity [3]. As such, they become susceptible to exploitable bugs that may only surface post-deployment.

A minor error in defining and coding the smart contract logic can introduce security vulnerabilities, potentially leading to substantial economic losses in the millions of dollars or incurring penalties. Examples of vulnerabilities in smart contracts include arithmetic bugs, exceptions, re-entrancy, and flash loan attacks [4]. To address the security concerns surrounding smart contracts, various tools have been developed employing both formal and non-formal techniques. Formal methods-based tools, such as Theorem provers like F* [5], Event-B, SMT Solvers [6], and proof assistants like FEther [7], Coq, Isabelle/HOL [8], enable rigorous verification of smart contract code to identify potential vulnerabilities. On the other hand, non-formal techniques-based tools, such as symbolic execution [9], data flow analysis [10], runtime monitoring [11], and fuzzing [12] etc., focus on dynamic analysis of smart contracts to detect vulnerabilities during execution.

However, formal verification tools, grounded in formal operational semantics, provide robust security guarantees at both pre and post smart contract deployment stages. These tools enable the formal specification and verification of contract properties, offering detection capabilities for a wide range of vulnerabilities. However, their complexity, steep learning curve, and intricate implementation make them challenging to use and automate. In contrast, non-formal methods, while simpler, focus on specific vulnerabilities, posing difficulties in extending to new vulnerability types. This limitation necessitates the deployment of multiple tools, leading to a cumbersome testbed setup for smart contract developers and users, as well as increased latency and hindered runtime analysis during vulnerability scanning.

Thus, recent years have seen a shift towards adopting machine learning (ML) to overcome the constraints of formal and non-formal methods for detecting bugs and flaws in smart contracts. ML enhances the speed and reliability of vulnerability detection, invoking substantial research interest in its application for smart contract verification within the research community. Previous ML studies, including the Long-Short-Term Memory Model by Tan et al. [13], Ensemble Learning-Based Smart Contract Vulnerability Prediction (SCVDIE-ENSEMBLE) by Zhang et al. [14], the Predictive Model by Momeni et al. [15], the Deep Learning-based approach by Nicolas et al. [16], and Graph Neural Networks technique by Zhuang et al. [17], among others. Notably, these techniques encounter common challenges, namely, they necessitate access to source code, available for only 1% of all contracts. Furthermore, they primarily differentiate between vulnerable and safe (no vulnerabilities) smart contracts, operating within a binary classification paradigm, lacking the capacity to identify specific vulnerability types. Additionally, they exhibit resource-intensive characteristics as those operating on the bytecode level and employing binary classification paradigms.

Therefore, in the realm of smart contract security, the detection of vulnerabilities remains a crucial task. To mitigate these challenges, this study proposes the utilization of lightweight convolutional neural networks (CNNs) for identifying vulnerabilities in Solidity smart contracts represented in RGB-encoded format. This innovative approach introduces several compelling advantages, including:

- *Reduced Computational Complexity:* Unlike traditional CNN models, which can be computationally demanding, lightweight CNNs employ a smaller number of parameters, significantly enhancing their computational efficiency. This characteristic makes them well-suited for deployment on resource-constrained devices, ensuring applicability in diverse settings.
- *Effective Pattern Recognition:* Lightweight CNNs excel at learning and recognizing intricate patterns in input images. This capability proves particularly valuable in the context

of smart contract vulnerability detection, as it enables the identification of subtle patterns that may indicate the presence of exploitable flaws.
- *Enhanced Scalability:* The combination of lightweight CNNs and RGB encoding facilitates rapid and scalable vulnerability scanning of deployed smart contracts. This is achieved through the streamlined assessment of thousands of contracts simultaneously, significantly surpassing the capabilities of traditional formal verification tools.

Furthermore, the transformation of smart contract bytecode into RGB-encoded images offers additional benefits:

- *Privacy Preservation:* RGB encoding effectively masks the proprietary information contained in smart contracts, safeguarding sensitive data while preserving the structural information essential for vulnerability detection.
- *Compactness and Efficient Storage:* RGB encoding transforms smart contracts into compact image representations, reducing storage requirements and enabling efficient storage on online platforms. This feature proves particularly advantageous for managing large repositories of smart contracts.

The proposed approach leverages the strengths of lightweight CNNs and RGB encoding to address the challenges of smart contract vulnerability detection. By reducing computational complexity, enhancing pattern recognition, and ensuring scalability, this methodology paves the way for a more secure and efficient smart contract ecosystem.

II. Lightweight CNN Models

This section provides a brief overview of lightweight Convolutional Neural Network (CNN) models, namely EfficientNetV2-S, EfficientNetV2-M, and MobileNetV3, considered for detecting fixed vulnerability patterns in smart contracts. The choice of MobileNetV3 is motivated by its specialization in lightweight and resource-efficient deployment on devices with limited resources, such as smartphones. Conversely, EfficientNetV2 is selected for its focus on optimizing training efficiency and parameter optimization without compromising overall performance quality. As smart contracts gain popularity in areas like supply chain management, finance, and healthcare, the need for efficient vulnerability detection on resource-constrained devices like IoT devices or smartphones becomes crucial. Fast detection is essential, given that a single smart contract may execute thousands or millions of times daily, preventing potential slowdowns.

A. EfficientNetV2

The deep learning models' training efficiency is crucial with the expansion of the model size and training data size. For example, Tom et al. [18], in their study, trained GPT-3 (an autoregressive language model) with 175 billion parameters, 10x more than any previous non-sparse language model, and tested its performance in the few-shot setting. It demonstrates the remarkable capability in few-shot learning, but the challenge arises from the fact that retraining or making improvements to such a model is a difficult task because it takes weeks of training with thousands of GPU cycles consumption. To address this issue, EfficientNet represents a convolutional neural network design coupled with a scaling method that uniformly fine-tunes the network's depth, width, and resolution using a combined coefficient. Meanwhile, within the EfficientNet family, the introduction of EfficientNetV2 marks a progression beyond the original architecture.

This network model is created through a fusion of scaling factors encompassing width, depth, and resolution and neural architecture search. The prime objective is to enhance training speed and parameter efficiency by incorporating a combination of MobileNetV2 inverted residual block (MBConv) and Fused MBConv to make the training faster without increasing parameters. EfficientNetV2 has several variants, each tailored to specific efficiency and performance considerations. These variants include EfficientNetV2-S, EfficientNetV2-M, and EfficientNetV2-L. According to Tan et al. EfficientNetV2 surpasses both prior and newer state-of-the-art models in terms of speed and efficiency. It trains up to 4x faster than prior models i.e., vision transformers, NFNets, and ConvNets etc., while being up to 6.8x smaller in parameter size.

However, in comparison to the Vision Transformer (ViT), EfficientNetv2-S flaunts a relative parameter count of around 24 million, and EfficientNetv2-M has 53.15 million parameters while the Vision Transformer (ViT) integrates roughly 86 million parameters. Moreover, the V2 iteration preserves nearly half of the parameters found in the original EfficientNet. Although it notably reduces the parameter size, it still manages to achieve similar or even higher levels of accuracy compared to other models when evaluated on the ImageNet dataset.

B. MobileNetV3

MobileNetV3 is categorized into two models: MobileNetV3-Large and MobileNetV3-Small. These models are designed for different scenarios, with MobileNetV3-Large intended for high-resource use cases and MobileNetV3-Small tailored for low-resource scenarios.

It is a lightweight convolutional neural network model that is optimized for mobile CPUs by integration of hardware-aware network architecture search (NAS) and the NetAdapt technique [19]. It is later enhanced by new architectural developments. Additions include (1) new efficient network design, (2) new efficient non-linearities suitable for mobile environments, and (3) complementary search algorithms. The network design modification involves adding the "hard swish" activation and squeeze-and-excitation modules in MBConv blocks. This reduces memory access, lowers latency costs, and boosts accuracy. Although parameter counts slightly increase, latency remains unaffected. [20]. However, in comparison to EfficientNet, it has a 2.99 million parameter count. Conclusively, EfficientNetV2 and MobileNetV3 are both lightweight CNN models that are well-suited for detecting fixed vulnerability patterns in smart contracts. EfficientNetV2 offers better performance, but it has a higher parameter count than MobileNetV3. MobileNetV3 is a preferable choice for applications where resources are limited.

III. Methodology

This study proposes a method for rapidly scanning fixed-pattern vulnerabilities in RGB-encoded smart contracts using lightweight convolutional neural networks (CNNs) for automatic feature extraction and learning, reducing the need for manual feature extraction, see Figure 1. This technique preserves complex information in the contract source code due to the approximately 16.8 million available colors in RGB images, as opposed to grayscale images with only 256 colors. RGB encoding may offer some advantages over text-based data in terms of compactness, such as size reductions that facilitate transmission and storage. Additionally, RGB images can be encrypted to protect privacy, making them ideal for applications where security and confidentiality are critical, such as secure voting systems, efficient financial transactions, supply chain management, and intellectual property protection.

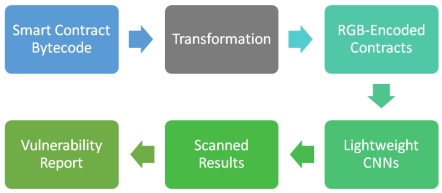

Fig. 1: System Architecture

A. Dataset

This study used the largest available dataset of Solidity smart contracts, provided by Martina et al. [21], which contains 100,000 smart contracts labeled using the Slither framework [10]. This dataset was divided into text and image classification tasks, but this study focused specifically on identifying fixed-pattern vulnerabilities in RGB-encoded contracts. Therefore, the approach only used image classification, where Solidity bytecode was converted into RGB color codes and transformed into fixed-sized encoded images. This method facilitates the detection and classification of code vulnerabilities. The dataset was partitioned into three splits: training $(79.6k)$, validation $(10.8k)$, and test $(15.9k)$. Multiple vulnerabilities may reside as part of a single contract.

B. Vulnerability Classes

The dataset has been categorized into five vulnerability classes for training and testing purposes:

1) Re-entrancy Vulnerability
2) Access Control Vulnerability
3) Integer overflow/underflow Vulnerability
4) Unchecked Calls Vulnerability
5) Miscellaneous Vulnerability (uninitialized-state variables, incorrect-equality, backdoor functions)

C. Architecture of Proposed System

Table I provides an overview of the architectural specifications for both Heavy and Light CNN models employed in this study for the purpose of smart contract vulnerability detection.

TABLE I: Model Architecture

Model (Heavy vs light CNNs)	Layers	Neurons	Kernel	Stride
ResNet-18 (H)	18	50, 100, 256, 512	3x3	2
Inception (H)	42	64, 192, 256, 288, 320, 480, 528	1x1, 3x3	1, 2
MobilenetV3-S (L)	56	16, 24, 32, 48, 64, 96, 160, 320	3x3	1, 2
EfficientNetV2-M (L)	67	24, 48, 96, 144, 240, 480, 960	3x3	1, 2

Whereas, Table II summarizes the hyperparameter settings employed in the study, including activation function, loss function, optimizer, epochs, metrics, batch size, learning rate, weight decay, and fine-tuning layers. The choice of the $ReLU$ activation function introduces non-linearity to enable the model to capture complex relationships. Categorical Crossentropy serves as the loss function, guiding the model to minimize errors in predicting class labels. Stochastic Gradient Descent (SGD) is employed as the optimization algorithm, iteratively adjusting model parameters to minimize the loss. Training occurs over 99 epochs, with evaluation metrics including both Accuracy and F1-Score for a comprehensive assessment of model performance. A batch size of 16 dictates the number of data samples processed in each iteration, while a learning rate of $1e-3$ determines the step size for parameter adjustments. Weight decay of 0.0001 introduces regularization to prevent overfitting. Notably, the fine-tuning of six layers allows leveraging pre-trained models, enhancing the adaptability of the models to the specific task at hand.

TABLE II: Hyperparameter Settings

Hyperparameter	Value
Activation Function	ReLU
Loss Function	Categorical Crossentropy
Optimizer	sgd
Epochs	99
Metrics	Accuracy & F1-Score
Batch-size	16
Learning rate	1e-3
Weight Decay	0.0001
Fine-tune Layers	6

D. Evaluation of Lightweight vs Heavy CNNs Models

A comprehensive evaluation of the proposed smart contract vulnerability detection technique was conducted using standard performance metrics, which are accuracy, precision, recall, and F1-score. The experimental results on RGB-encoded contract vulnerability detection, employing a fixed set of vulnerability patterns, revealed that Lightweight CNN models, namely EfficientNetV2 and MobileNetV3, outperformed ResNet-18 and Inception-v3. Figure 2 illustrates that the EfficientNetV2 model achieved the highest mean F1 score (0.78), followed by MobileNetV3 (0.76), ResNet (0.73), and Inception (0.67). Furthermore, the EfficientNetV2 model demonstrated the highest mean recall score (0.74) and mean precision score (0.83), solidifying its position as the top performer for vulnerability

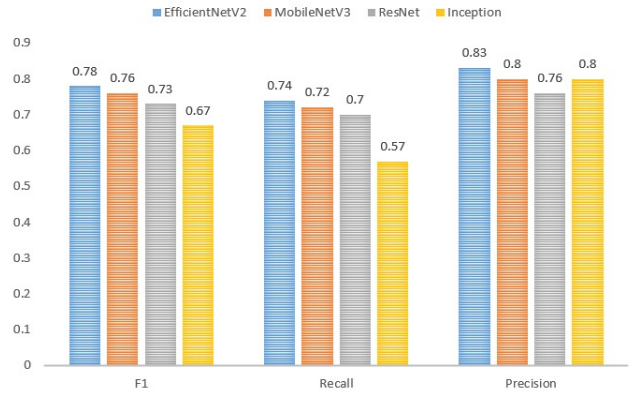

Fig. 2: Comparative Analysis: Light vs Heavy CNNs

detection in RGB-encoded images. In Figure 3, a detailed analysis has been presented, contrasting the performance of EfficientNetV2 and MobileNetv3 with ResNet (the model identified as the best for vulnerability detection in the study by Martina et al. [21]). To ensure a fair and consistent comparison, identical training and validation configurations were applied to all the models using the dataset [21]. The findings revealed that both EfficientNetV2 and MobileNetv3 exhibited remarkable stability, maintaining a close alignment between training and validation accuracy, with a difference of approximately 1%, see Figure 3. This suggests that both models effectively generalize to unseen data. In contrast, ResNet displayed a significant gap of approximately 31% between training and validation accuracy, indicating a susceptibility to overfitting.

The observed performance difference between our best model, EfficientNetV2 and ResNet can be attributed to the underlying architectural designs of the two models. EfficientNetV2 incorporates efficient scaling and regularization techniques that promote robust learning and prevent overfitting. Conversely, ResNet's architecture, while effective in certain contexts, may be more prone to overfitting, especially when dealing with large and complex datasets as in this case. Hence, the findings of the proposed study underscore EfficientNetV2's superior generalization ability, making it a compelling choice for rapid and reliable fixed pattern vulnerability detection in RGB-encoded images, particularly when dealing with large datasets.

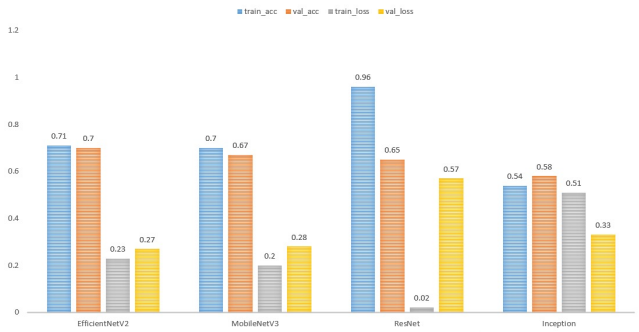

Fig. 3: Model Performance: Accuracy vs Loss Comparison

IV. Conclusion

This study explored the use of lightweight CNNs for the efficient detection of vulnerability patterns in Solidity RGB-encoded smart contracts. The exploration of models such as EfficientNetV2 and MobileNetV3 showcased superior performance compared to heavy counterparts like ResNet-18 and Inception-v3. EfficientNetV2, in particular, emerged as the top performer with the highest mean F1 score (0.78) and demonstrated stability in both training and validation accuracy. This approach not only reinforces security and protects proprietary information but also addresses compactness concerns, facilitating convenient storage on online platforms. The methodology ensures an efficient use of bandwidth, enabling rapid post-deployment scanning of contracts for potential vulnerabilities.

References

[1] Sendner, Christoph, et al. "Smarter Contracts: Detecting Vulnerabilities in Smart Contracts with Deep Transfer Learning." NDSS. 2023.
[2] Mustafa, Iqra, et al. "Decentralized Oracle Networks (DONs) Provision for DAML Smart Contracts." International Congress on Blockchain and Applications. Cham: Springer Nature Switzerland, 2023.
[3] Beregszaszi, C. Alex. "The Solidity Contract-Oriented Programming Language." Accessed (2020–07-08.) (2020).
[4] Kaur, Gurdip, et al. "Smart Contracts and DeFi Security and Threats." Understanding Cybersecurity Management in Decentralized Finance: Challenges, Strategies, and Trends. Cham: Springer International Publishing, 2023. 91-111.
[5] Bhargavan , K. et al., "Formal Verification of Smart Contracts: Short Paper," Proceedings of the ACM Workshop on Programming Languages and Analysis for Security, Vienna, Austria, 2016.
[6] Alt, Leonardo, and Christian Reitwiessner. "SMT-based verification of solidity smart contracts." Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice: 8th International Symposium, ISoLA 2018, Limassol, Cyprus, November 5-9, 2018, Proceedings, Part IV 8. Springer International Publishing, 2018.
[7] Z. Yang and H. Lei, "Fether: An extensible definitional interpreter for smart-contract verifications in coq." IEEE Access, 2019.
[8] Amani, Sidney, et al. "Towards verifying ethereum smart contract bytecode in Isabelle/HOL." Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs. 2018.
[9] Luu, Loi, et al. "Making smart contracts smarter." Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. 2016.
[10] Feist, Josselin, Gustavo Grieco, and Alex Groce. "Slither: a static analysis framework for smart contracts." 2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB). IEEE, 2019.
[11] Cook, Thomas, Alex Latham, and Jae Hyung Lee. "Dappguard: Active monitoring and defense for solidity smart contracts." Retrieved July 18 (2017): 2018.
[12] Fu, Ying, et al. "Evmfuzzer: detect evm vulnerabilities via fuzz testing." Proceedings of the 2019 27th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering. 2019.
[13] Tann, Wesley Joon-Wie, et al. "Towards Safer Smart Contracts: A Sequence Learning Approach to Detecting Security Threats." arXiv preprint arXiv:1811.06632 (2018).
[14] Zhang, Lejun, et al. "A novel smart contract vulnerability detection method based on information graph and ensemble learning." Sensors 22.9 (2022): 3581.
[15] Momeni, Pouyan, Yu Wang, and Reza Samavi. "Machine learning model for smart contracts security analysis." 2019 17th International Conference on Privacy, Security and Trust (PST). IEEE, 2019.
[16] Lesimple, Nicolas, and Martin Jaggi. Exploring deep learning models for vulnerabilities detection in smart contracts. Diss. Master's thesis, Massachusetts Institute of Technology, 2020.
[17] Zhuang, Yuan, et al. "Smart Contract Vulnerability Detection Using Graph Neural Networks., 2020"
[18] Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.

[19] Yang, Tien-Ju, et al. "Netadapt: Platform-aware neural network adaptation for mobile applications." Proceedings of the European Conference on Computer Vision (ECCV). 2018.
[20] Howard, Andrew, et al. "Searching for mobilenetv3." Proceedings of the IEEE/CVF international conference on computer vision. 2019.
[21] Rossini, Martina, Mirko Zichichi, and Stefano Ferretti. "Smart Contracts Vulnerability Classification through Deep Learning." Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems. 2022.