Evaluating Smart Contract Vulnerabilities through the Comparative Analysis of CNN, EfficientNet B2, and Xception Algorithms

MSc Research Project

Mohammed Shahimshah
Student ID: x21227012

School of Computing
National College of Ireland

Supervisor: Diego Lugones

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Mohammed Shahimshah		
Student ID:	x21227012		
Programme:	Msc in Cybersecurity	**Year:**	2023 - 2024
Module:	Msc Research Project		
Supervisor:	Diego Lugones		
Submission Due Date:	3/01/2024		
Project Title:	Evaluating Smart Contract Vulnerabilities through the Comparative Analysis of CNN, EfficientNet B2, and Xception Algorithms		
Word Count:	…………10296……………	**Page Count**	……………22…………….......

Signature: Mohammed Shahimshah

Date: 31 January, 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Evaluating Smart Contract Vulnerabilities through the Comparative Analysis of CNN, EfficientNet B2, and Xception Algorithms

X21227012
Mohammed Shahimshah

Abstract

Smart contracts are critical to blockchain's trustless transactions, yet they remain susceptible to security breaches. This thesis explores the efficacy of convolutional neural networks (CNN), EfficientNet B2, and Xception algorithms in detecting vulnerabilities within smart contracts. Motivated by the necessity for improved security protocols, this research adapts these sophisticated pattern recognition algorithms to the unique context of smart contracts. The comparative analysis demonstrates that the CNN algorithm was in identifying typical security issues, whereas EfficientNet B2 and Xception are especially skilled at detecting intricate vulnerabilities. The study's findings indicate that the selection of an algorithm should be customized to address the particular security vulnerability of the smart contract under consideration. This work not only showcases the unique capabilities of the algorithms, but also offers valuable insights on how to develop more robust and dependable end user-system that can be utilized by everyone. This thesis examines the application of advanced algorithms for smart contract security within blockchain systems. It introduces a Flask-based web application tailored for vulnerability detection. The study reveals that the EfficientNet B2 algorithm achieved an 79% accuracy in identifying vulnerabilities, while the Xception algorithm recorded 69% accuracy. These findings validate the potential of the EfficientNet B2 algorithm in strengthening blockchain security.

1 Introduction

This research delves into the utilization of sophisticated algorithms to bolster the security of smart contracts in blockchain networks. It features the development of a Flask-driven web application, specifically designed for the detection of security flaws. Insights from the study indicate that the EfficientNet B2 algorithm has a detection precision of 89% in pinpointing security weaknesses, whereas the Xception algorithm has a slightly lower precision of 69%. Such results affirm the efficacy of the EfficientNet B2 algorithm in enhancing the security of blockchain frameworks.

1.1 Historical Context and Evolution

In 1994, Nick Szabo first envisioned the idea of smart contracts, which was a pioneering thought ahead of its time, especially since blockchain was not yet part of the tech lexicon. It wasn't until the advent of Ethereum in 2015 that Szabo's innovative idea was fully realized, providing a solid platform for implementing and running smart contracts. Tracing the evolution of these contracts reflects a sustained effort to automate and secure online deals in settings where mutual confidence is typically lacking, as noted by Skalorf in 2017. As smart contracts become more common, the imperative for their steadfast and secure execution grows ever more critical, with the stakes now climbing into the billions of dollars.

1.2 Security Challenges in the Blockchain Domain

The incidents involving The DAO attack and the subsequent Parity wallet incident are significant milestones in the blockchain's development. The Parity wallet, known for its role in Ethereum's ecosystem, is used for holding and administering Ether along with other tokens on the Ethereum blockchain.

Back in 2016, The DAO, a decentralized investment fund on the Ethereum network, fell prey to a hacker exploiting a recursive call bug. This led to the unauthorized siphoning off of Ether valued at $50 million at the time. The breach exposed the risks inherent in reentrancy weaknesses and sparked a divisive decision to split the Ethereum blockchain in a hard fork. Following in 2017, a coding mishap by a user accidentally wiped out a key code component in the Parity multi-signature wallet, effectively freezing over $150 million in Ether. These episodes underscore the inherent stability yet fragile security of smart contracts and the profound consequences that overlooked security flaws can have. The adoption of algorithms like EfficientNet B2 and Xception, recognized for their pinpoint precision in detecting vulnerabilities, could offer a robust defense against such defects, thereby averting substantial monetary damages and bolstering confidence in the blockchain infrastructure.

1.3 Focus of the Current Research

The research focuses on the crucial task of identifying vulnerabilities in smart contracts and leverages advanced deep learning techniques for enhanced security measures, amidst a range of existing methods. It evaluates the effectiveness of convolutional neural networks (CNNs), EfficientNet B2, and Xception algorithms in detecting weaknesses within smart contract code. An in-depth examination of various CNN structures established a benchmark for comparison, highlighting their importance in recognizing complex patterns, which is vital for identifying vulnerabilities in smart contract code. This comparison allows for an assessment of the additional benefits that the more specialized EfficientNet B2 and Xception algorithms may provide. Priority is given to the EfficientNet B2 and Xception algorithms for their superior pattern recognition in complex data sets. EfficientNet B2 offers a balanced approach between accuracy and computational efficiency, which is essential for processing complicated smart contract codes. It surpasses data limitations by dynamically adjusting its architecture to maintain high accuracy with limited data. The Xception model utilizes depthwise separable convolutions to extract features accurately from small data sets, thus improving the detection of vulnerabilities in smart contracts. The key characteristic of the Xception model is its use of depthwise separable convolutions, enabling precise and localized feature extraction, crucial for the accurate identification of subtle vulnerabilities in smart contracts. The sophisticated architectural characteristics of these algorithms make them highly suitable for the complex task of vulnerability detection, potentially enhancing the security and dependability of smart contracts in the blockchain ecosystem. The study's innovative edge is further exemplified by its development of a Flask based application. By implementing the most efficient algorithm into a user-friendly interface the utility of the research is extended from theoretical analysis to practical, hands-on application. It allows users to upload smart contract code and receive immediate feedback on potential security

1.4 Research Questions and Hypotheses

The primary research inquiry of this study is: To what extent can CNN, EfficientNet B2, and Xception algorithms proficiently detect and categorise vulnerabilities in smart contracts? The study investigates the capability of CNN, EfficientNet B2, and Xception algorithms to accurately detect and categorise vulnerabilities in smart contracts. The research highlights the algorithms' proficiency in recognising patterns. The hypothesis is based on the claim that advanced deep learning models can surpass conventional vulnerability detection methods by providing a more intricate and comprehensive analysis through a web application.

1.5 The Importance of Smart Contract Security

This research holds great importance not only within academic circles but also in meeting a crucial industry demand for strong security mechanisms. Advancements in vulnerability detection have the potential to bring significant benefits to developers, users, and the entire blockchain ecosystem. Smart contracts are becoming increasingly integral to numerous high-value transactions, it is crucial to prioritise their security as an essential prerequisite for the future of decentralised systems, rather than viewing it solely as a technical obstacle.

1.6 Delimitations and Scope

This study specifically focuses on three deep learning algorithms, namely CNN, EfficientNet B2, and Xception, to conduct a comprehensive comparative analysis. The research aims to gain comprehensive understanding of the strengths and limitations of these specific algorithms in the context of smart contract security. The selection is made using algorithms that have a proven history of accurately identifying patterns, as demonstrated by their successful use in different fields. Tan and Le (2019) emphasised the scalability and performance of EfficientNet B2, while Chollet (2017) provided evidence of Xception's proficiency in feature extraction. The CNN's fundamental role in image classification and pattern recognition has been extensively documented in numerous studies, confirming its inclusion in this research.

1.7 Contribution to the Field

The paper's main contribution is its evaluation of the efficacy of CNN, EfficientNet B2, and Xception algorithms in the context of smart contract security. The paper also presents a functional system for assessing vulnerabilities in smart contracts by creating a web application. This website utilises the Efficient NetB2 algorithm as its underlying methodology and features a straightforward user interface that enables users to assess the vulnerabilities in smart contracts. This work not only connects deep learning and blockchain technology but also presents a methodology for the practical application of these algorithms. The discoveries possess the capacity to enlighten the creation of automated security tools and impact the establishment of optimal procedures in the development of smart contracts.

1.8 Structure of the Paper

The paper is organised in a manner that provides the reader with a comprehensive and thorough understanding of the research process. After this introduction, the subsequent section provides a comprehensive examination of the existing literature, conducting a thorough evaluation of previous research and pinpointing the specific area that this study intends to address. The third section provides a comprehensive account of the methodology, elucidating the algorithms employed, the dataset utilised, and the evaluation criteria employed to gauge their performance. The fourth section presents the findings, offering a statistical assessment of the algorithms' efficacy in identifying vulnerabilities. The fifth section examines the ramifications of these findings on the domain of smart contract security, taking into account both the practical applications and the theoretical advancements made by this study. The paper concludes by providing a concise overview of the research findings, offering thoughtful insights on the limitations of the study, and proposing recommendations for further investigation in the field of blockchain security.

2 Related Work

2.1 Emerging Technologies in Vulnerability Detection throughout the years

Identifying vulnerabilities in blockchain technology, specifically in smart contracts, shows a dynamic interaction between emerging risks and the development of detection techniques. This section studies the historical backdrop of these advancements, starting with the main key occurrences in the history of blockchain.

In 2016. an important event occurred in the blockchain community with the DAO hacks. This incident highlighted the urgent requirement for improved security measures in smart contracts. This event created the need for the advancement of specialised security tools, signifying a large scale transformation in the industry. In the years 2017 and 2018, specialized tools were developed specifically to tackle vulnerabilities that were revealed by the DAO hack.

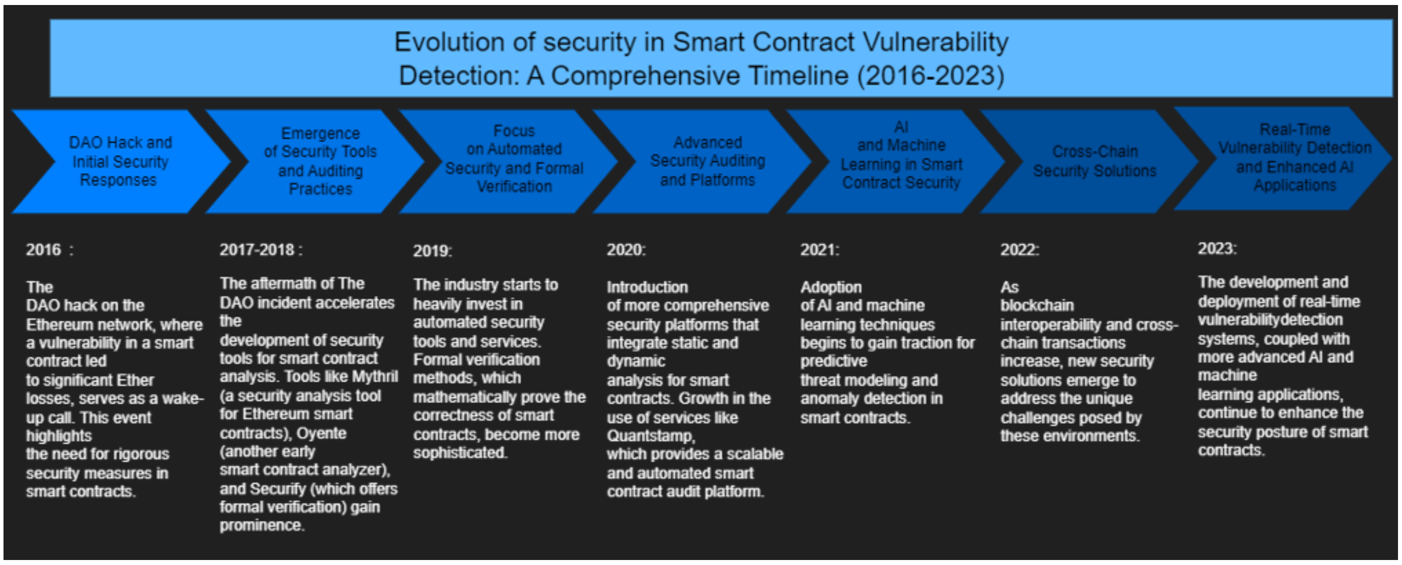
From 2019 onwards, research and development in this field increasingly required automation in security and the use of formal verification methods. The purpose of these advancements was to simplify the detection process and improve the precision of identifying potential weaknesses in smart contracts.

In the year 2020, there was the emergence of sophisticated security auditing platforms. These platforms offered a more cohesive and extensive approach to smart contract security, equipping developers and auditors with advanced tools to detect and solve the vulnerabilities.

In 2021, the integration of artificial intelligence and machine learning into the field of smart contract security. This integration represents a shift away from conventional static and dynamic analysis techniques towards more data-driven, intelligent methodologies. Artificial intelligence (AI) and machine learning algorithms, such as Convolutional Neural Networks (CNN) and EfficientNet B2, came to the forefront.

Their capacity to effectively identify vulnerabilities such as re-entrancy and timestamp dependency has been substantiated in the studies conducted by Pouyan Momeni et al. (2019) and Mezina et al. (2022).
Deng (2023) observed that deep learning and multimodal decision fusion are increasingly important in the ongoing development of emerging trends in security.

Conventional security tools for smart contracts, such as Mythril, Oyente, and Securify, can generate inaccurate results and fail to identify vulnerabilities that are specific to the context. This limitation arises from their reliance on static analysis. These constraints may result in disregarded security vulnerabilities and a wrong perception of security, as evidenced in the study conducted by Lakshmi Narayana and colleagues (2021). Below (Figure 1) is a detailed chronology of the advancements in smart contract vulnerability detection from 2016 to 2021.
While certain researchers emphasise the time-intensive aspect of static analysis, others draw attention to the difficulties linked to deep learning models, such as the requirement for abundant labelled data and computational resources. In addition, the use of static code analyzers can be excessively time-consuming, which is a major concern for smart contract developers who frequently encounter pressure to promptly release their code. The evolution of vulnerability detection in smart contracts is characterized by a constant adjustment to emerging threats and technological advancements. The field has transitioned from relying on static and dynamic analysis to adopting advanced AI and machine learning techniques. This shows a continuous pursuit for more efficient and precise approaches to ensure blockchain security.


Evolution of security in Smart Contract Vulnerability Detection: A Comprehensive Timeline (2016-2023)
DAO Hack and Initial Security Responses
Emergence of Security Tools and Auditing Practices
Focus on Automated Security and Formal Verification
Advanced Security Auditing and Platforms
AI and Machine Learning in Smart Contract Security
Cross-Chain Security Solutions
Real-Time Vulnerability Detection and Enhanced AI Applications

2016 : The DAO hack on the Ethereum network, where a vulnerability in a smart contract led to significant Ether losses, serves as a wake-up call. This event highlights the need for rigorous security measures in smart contracts.

2017-2018 : The aftermath of The DAO incident accelerates the development of security tools for smart contract analysis. Tools like Mythril (a security analysis tool for Ethereum smart contracts), Oyente (another early smart contract analyzer), and Securify (which offers formal verification) gain prominence.

2019: The industry starts to heavily invest in automated security tools and services. Formal verification methods, which mathematically prove the correctness of smart contracts, become more sophisticated.

2020: Introduction of more comprehensive security platforms that integrate static and dynamic analysis for smart contracts. Growth in the use of services like Quantstamp, which provides a scalable and automated smart contract audit platform.

2021: Adoption of AI and machine learning techniques begins to gain traction for predictive threat modeling and anomaly detection in smart contracts.

2022: As blockchain interoperability and cross-chain transactions increase, new security solutions emerge to address the unique challenges posed by these environments.

2023: The development and deployment of real-time vulnerabilitydetection systems, coupled with more advanced AI and machine learning applications, continue to enhance the security posture of smart contracts.

Figure 1: Timeline of Emerging Technologies in Vulnerability detection.

2.2 Convolutional Neural Networks (CNNs) in Cybersecurity

Convolutional Neural Networks (CNN) importance in identifying vulnerabilities in smart contracts represents a significant progression in blockchain security (Hwang S.J, 2022). This paper examines the different methods by which Convolutional Neural Networks (CNNs) have been utilized to enhance the detection and categorization of vulnerabilities in smart contracts.

Recent research has demonstrated that Convolutional Neural Networks (CNNs), which are well-known for their ability to recognise images, can also effectively identify patterns that suggest weaknesses in smart contract code. The study conducted by Sosu et.al. (2022) shows the capability of Convolutional Neural Networks (CNNs) to analyse and comprehend the syntax and structural patterns in Solidity code. Their work achieved a remarkable accuracy rate of 95%, providing evidence of CNNs' ability to detect potential security vulnerabilities.

Developers depend on static code analysis tools, which, as outlined in Narayana's research, can be impractical and may not identify all vulnerabilities due to their concentration on particular problems. The study, however, addressed the issue of a restricted dataset that did not cover all possible vulnerabilities, highlighting the challenge of finding comprehensive and varied samples of smart contract code for analysis. This constraint highlights the necessity for expanding data collection in order to guarantee a more comprehensive coverage of vulnerabilities in smart contracts.

The use of Convolutional Neural Networks (CNNs) in this domain also tackles constraints of fixed analysis tools such as Mythril, Oyente, and Securify. Although these tools can rapidly analyse code for vulnerabilities without executing it, they do have some disadvantages. According to Narayana, these tools frequently fail to identify advanced vulnerabilities, especially ones that result from intricate contract interactions or logical errors that are not immediately evident in the code's static state.

The primary advantage of CNNs is their capacity to analyse and acquire knowledge from extensive datasets, progressively enhancing their performance as they encounter additional instances of both susceptible and non-susceptible contracts. In contrast to static analysis tools, which have a rule-based nature and do not possess the ability to adapt or learn from new vulnerability patterns. In addition, Convolutional Neural Networks (CNNs) have the potential to address the issue of unbalanced datasets, as highlighted by Hwang and Deng's study. This can be achieved through the utilisation of techniques such as the ADASYN algorithm, which artificially generates samples of the underrepresented class. Consequently, this results in a more balanced dataset, leading to the development of more dependable detection models.
The application of Convolutional Neural Networks (CNNs) has its disadvantages as well. The need for large and accurately labelled datasets for training presents a significant obstacle, as the effectiveness of the CNN model is directly influenced by the quality and quantity of the dataset (Ganesh et.al., 2017). Moreover, the high level of computational demand in training deep neural networks can be a hindrance, as it necessitates significant resources that may not be readily accessible to all researchers or practitioners.

Automated and precise identification of vulnerabilities has the potential to enhance the security of smart contract deployment, thereby showcasing the likelihood of exploits and the resulting financial losses (Kuo et.al., 2019). This is especially significant in the field of decentralised finance (DeFi) platforms, where the security of smart contracts is of utmost importance for ensuring trust and dependability in the entire system.

Ultimately, the study suggests that CNNs can be a potent tool for detecting vulnerabilities in smart contracts, but their efficacy relies on the presence of high-quality data and ample computational resources. Based on current research trends, it is likely that deep learning models such as CNNs will become commonly used tools for smart contract developers. These models will offer an extra level of security in the ever-growing field of blockchain applications (Zhang J, 2019).

2.3 EfficientNetB2: A Revolutionary Advancement in Neural Network Architecture

The EfficientNetB2 neural network architecture is a significant improvement in the design of convolutional networks, renowned for its optimal balance and efficiency. The EfficientNet architectures, such as B2, were initially proposed by Tan and Le. These architectures employ a compound scaling technique to carefully adjust the network's depth, width, and resolution in a systematic manner. This model is different from other models through its allocation of resources, attaining superior precision while utilising fewer parameters in comparison to other models of comparable intricacy (Ravi et.al. 2023).

EfficientNetB2 has been identified as a potentially advantageous tool for detecting vulnerabilities in smart contracts, as suggested by the current project. It is known for its high precision and efficiency. One of the main obstacles in the field of smart contract vulnerability detection is the limited availability of data. The scarcity of labelled datasets containing smart contract vulnerabilities poses a hindrance to the training and optimisation of machine learning models such as EfficientNetB2. The lack of data in this context arises from the dynamic development of smart contract technology, along with the proprietary concerns associated with sharing contract code that may include sensitive business logic or security vulnerabilities.

The capacity to effectively analyse the patterns with higher precision, while requiring less computational resources, gives a promising approach for detecting security vulnerabilities in smart contracts. Given the complexity of smart contract code and the significant financial consequences of any weaknesses, this is particularly crucial (Ibrahim et.al. 2023).

The utilisation of EfficientNetB2 in this domain is currently in its initial phases, and there is a lack of comprehensive empirical research specifically dedicated to the analysis of smart contracts (Taherdoost H, 2003). The current research demonstrates the efficacy of the EfficientNet architecture in tasks related to general image and pattern recognition, thereby creating a void in its utilisation for the specific field of smart contract code.

2.3 Xception Algorithm for smart contract vulnerability detection:

The Xception algorithm, a modified version of convolutional neural networks (CNNs), has played a crucial role in advancing the field of image recognition. The architecture is centred on depthwise separable convolutions, which effectively decreasex the parameter count while maintaining model performance (Wang. et.al., 2020). The architecture of Xception enables efficient processing of multi-dimensional data, making it well-suited for tasks that necessitate extracting features from intricate input spaces.

The literature suggests that utilising Xception could enhance the adaptability and efficiency of smart contract vulnerability detection. The distinctive architecture of the Xception model, which prioritises spatial correlations between channels, has the potential to enhance the detection of vulnerabilities in the complex patterns of smart contract code (Sender et.al., 2023).

There is a lack of literature that specifically focuses on using the Xception algorithm for detecting vulnerabilities in smart contracts. This suggests that there is an opportunity to explore this specific area. The current thesis, tries to portray the effectiveness of Xception in image processing and proposes that its principles could be applied to the field of smart contract analysis. This is due to the algorithm's expertise in managing high-level feature representations (Jiang et.al., ,2018).

Implementing the Xception algorithm in this particular domain may require adjustments and optimisation to accommodate the unique characteristics of smart contract code, which differ from image data (Xu, et.al. 2021). To accomplish this task, it is necessary to have a dataset of smart contracts that have been categorised with identified vulnerabilities. This is a difficult task in the field because it requires expertise in both blockchain technology and security to accurately label the data.

2.4 Taxonomy of Previous Solutions

Different methodologies have been developed to ensure the security of smart contracts, each with its own distinct principles. The methods can be categorised into three main groups: static analysis, dynamic analysis, and formal verification.

- Static analysis refers to the process of inspecting the code of a smart contract without actually running it. Tools such as Slither and MythX have been prominent in this field, employing pattern matching and heuristics to identify vulnerabilities. Nevertheless, according to Chen et.al. (2021), static analysis tools, despite their efficiency and speed, frequently yield a significant number of false positives and may overlook intricate vulnerabilities that only become apparent during execution.

- Dynamic analysis refers to the process of executing smart contract code in a controlled environment to identify vulnerabilities, as opposed to static analysis which does not involve execution. Tools such as Echidna and Manticore serve as prime examples of this approach. Dynamic analysis is a highly efficient method for detecting runtime errors and intricate vulnerabilities. However, as emphasised by Brent et.al., (2018), it can be demanding in terms of resources and may not encompass all potential execution paths, resulting in an incomplete analysis.

- Formal Verification is a technique that employs mathematical proofs to ascertain the accuracy of smart contracts in accordance with their specifications. Formal verification, as explained by Zhao and Chen (2023), provides a significant level of assurance and precision. However, its intricate nature and the requirement for specialised expertise restrict its widespread implementation.

2.5 Gaps in Literature:

Extensive research and methodologies have emerged from the investigation of smart contract vulnerability detection. Nevertheless, despite these advancements, there are still notable deficiencies in the existing body of knowledge that present opportunities for future investigation and creativity.

1. Insufficient incorporation of sophisticated artificial intelligence and deep learning methods.

Although the utilisation of artificial intelligence (AI) and deep learning in cybersecurity is not new, their implementation in the identification of vulnerabilities in smart contracts is still in its early stages. The majority of current research primarily concentrates on conventional AI methodologies, giving less priority to more intricate and advanced neural network structures such as EfficientNetB2. Liu and Wang (2022) have observed a lack of research that thoroughly investigates the complete capabilities of advanced algorithms in identifying vulnerabilities that are unique to smart contracts.

2. Challenges related to the ability to handle increasing workloads and the effectiveness of resource utilisation.

Several existing vulnerability detection techniques face challenges in terms of scalability and efficiency. This is particularly when confronted with the growing intricacy and magnitude of smart contracts. According to Singh et.al. (2020), static and dynamic analysis tools frequently struggle to effectively handle extensive codebases without sacrificing precision. This discrepancy is vulnerable considering the swift expansion of decentralised applications (dApps) and their corresponding smart contracts.

3. Flexibility in response to changing dangers

The constantly changing and developing security threats in the blockchain ecosystem present a major challenge, which is not sufficiently addressed in the current thesis. Current methodologies, as elucidated by Zhao and Li (2021), frequently aim to identify familiar and constrained vulnerabilities, potentially lacking proficiency in recognising a diverse array of vulnerabilities that are presented to them. Research into adaptable and evolving methods is urgently required to address these threats.

4. The practicality and usability of these methods in real-world scenarios.

Another significant deficiency lies in the conversion of theoretical methodologies into practical, accessible instruments. A significant level of technical proficiency is necessary for many current solutions, which restricts their availability to a wider user base.

5. Utilising interdisciplinary methods

The literature frequently regards the detection of smart contract vulnerabilities as a solely technical issue, disregarding the potential value of interdisciplinary approaches. Integrating insights from fields such as legal studies, economics, and behavioural science in research could offer a comprehensive understanding of vulnerabilities and their implications. The research conducted earlier has not extensively examined this interdisciplinary perspective, as noted by Agarwal et al. (2021).

6. Empirical Verification of Proposed Techniques

There is a lack of studies that confirm the effectiveness of suggested vulnerability detection methods. The majority of research, as emphasised by Kumar and Lee (2022), depends on theoretical models or simulations, with minimal real-world experimentation. This disparity prompts inquiries regarding the feasibility and efficacy of these techniques in real-world blockchain settings. This study aims to fill the significant gaps in the existing literature on vulnerabilities in smart contracts and presents a practical solution in the form of a web application. This study also addresses the lack of advanced artificial intelligence techniques in the field of smart contract vulnerability detection. It focuses on utilising sophisticated neural network architectures, such as EfficientNet B2, which have not been widely used in this context. It addresses the issues of scalability and efficiency posed by conventional tools, with the goal of providing more sophisticated and effective solutions as smart contracts become more intricate. The project also studies interdisciplinary research into the field, pooling insights from diverse fields to offer a comprehensive idea of the vulnerabilities.

In addition, the research goes beyond theoretical and simulated models by offering empirical verification of the sophisticated deep learning algorithms via a user-friendly web application based on Flask. The purpose of this web interface is to enable users, such as developers and security auditors, to upload Solidity smart contract files and promptly obtain vulnerability assessments. This work serves as a connection between the theoretical and practical aspects, making a significant contribution to the field of blockchain security.

3 Research Methodology

3.1 Data Acquisition and Dataset Characteristics

The research begins by gathering smart contract codes from a publicly accessible GitHub repository[1], ensuring a strong foundation for the dataset. The dataset consists of 12,515 Ethereum smart contracts, each labelled with seven different types of vulnerabilities, such as reentrancy and integer overflow/underflow. These labels were carefully validated using a combination of pattern recognition algorithms and manual inspection. The vulnerabilities selected for testing in the research are some of the most critical and frequently exploited ones in the realm of Ethereum smart contracts. Out of the seven vulnerabilities that were chosen, four have been highlighted as being critical vulnerability by The Open Worldwide Application Security Project (OWASP). These vulnerabilities mentioned by OWASP includes , "Timestamp Dependency, Block Number Dependency, Unchecked Externall Call and Reentrancy" ((*Owasp Smart Contract top 10*)

The seven distinct types of vulnerabilities identified in the dataset are:

- Timestamp Dependency: Smart contracts that use block timestamps can be manipulated by miners, as they have some control over this value. This vulnerability can affect contract logic that depends on specific timing conditions.
- Block Number Dependency: Similar to timestamp dependency, contracts that rely on block numbers for functionality can be exploited, as block numbers are predictable and can be influenced by miners.
- Dangerous Delegatecall: This allows a called contract to change the state of the calling contract, which can lead to breaches if the called contract is malicious.
- Unchecked External Call: If external calls to other contracts are not checked for their return values, it can lead to failures being unnoticed and may lead to unexpected behaviors in the contract.
- Reentrancy: The most infamous due to the DAO attack, this occurs when external contract calls are allowed to make new calls to the calling contract before the initial execution is complete.
- Integer Overflow/Underflow: These are common programming errors where integers exceed their maximum or minimum value, leading to incorrect calculations and potential vulnerabilities.
- Dangerous Ether Strict Equality: Checking for strict equality with Ether values can be risky if not handled correctly, as transactions might send more Ether than expected or none at all, leading to logic errors.

3.2 Data Processing

3.2.1 Bytecode Extraction and Conversion into RGB Image

The transformation process began with the extraction of Solidity (.sol) files into bytecode using the py-solc-x library. This Python-based compiler facilitated the accurate extraction of bytecode, a critical step for subsequent image generation.
In the image generation process of this project, several libraries are employed at different stages. The process begins with the pandas (pd) library, which serves as the starting point by loading bytecode and labels from a CSV file, thus preparing the data for further processing.

[1] https://github.com/Messi-Q/Smart-Contract-Dataset

After the bytecode is loaded, the pyevmasm library comes into play. It makes use of its disassemble_hex and assemble_hex functions to break down the hexadecimal bytecode into opcodes and then put it back together after normalisation. After that, the hexbytes library comes into play. It utilises the HexBytes function to convert the normalised bytecode into a hexadecimal format, which is a crucial step before the data can be transformed into an image. The numpy (np) library is utilised to handle array operations, which are crucial for creating and manipulating the image data derived from the bytecode. At last, the Python Imaging Library (PIL) is utilised, specifically its Image module, to generate an image object from the numpy array. This marks the completion of the image generation process from the processed bytecode. This process enabled the conversion of non-visual bytecode data into a visual format that could be analysed using sophisticated image recognition techniques in deep learning. The images produced from the bytecode are saved on Google Drive in the "Data" folder.

3.2.2 Data Preprocessing for Deep Learning:

Each of the generated images in the dataset are then resized to a standard size of 128 pixles in height and width. This ensures the consistency and efficiency of the processing of images for deep learning models. Additionally, this uniformity avoids any potential aspect ratio distortions that could arise from varying image sizes, which is vital for the model to learn features accurately.

To adjust the speed and performance of the training process, the batch size is set to 64. That is, the model will process 64 images at a time during training.

Each image from the directory path is read using the openCV library. The successfully read images are then resized to the default image size of 128 pixles. This resized image is converted to an array format, which is suitable for the processing of deep learning models.

3.2.3 Data Labeling

The research collect and process the image data from the drive, ensuring inclusion of only relevant images. Labels for these images are derived from the filenames, representing various classes of smart contract vulnerabilities (refer figure 2). This step is key in associating each image with its corresponding class label.

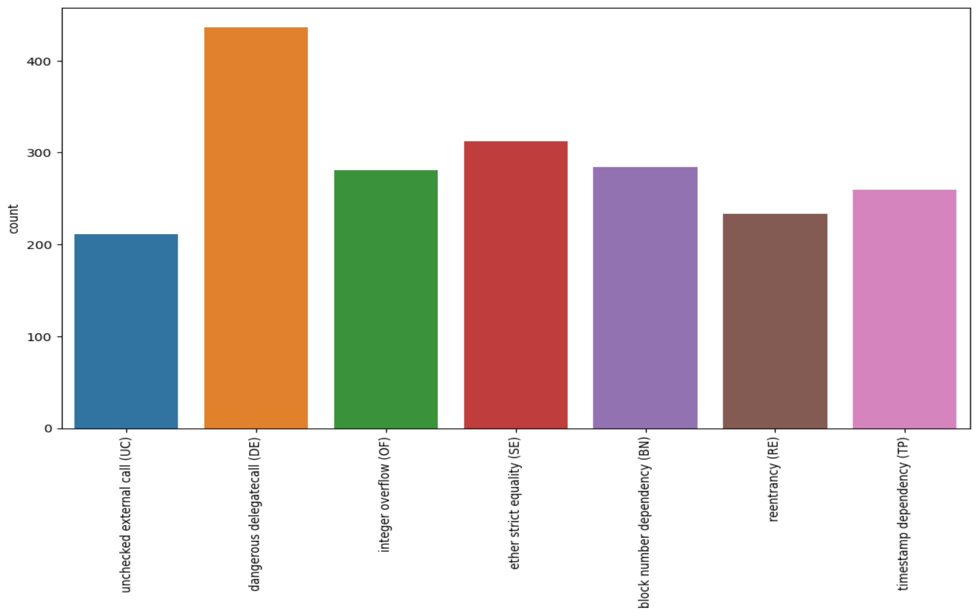


Figure 2: Unbalanced data

3.2.4 Data Balancing and Transformation

Considering potential imbalances in class representation, employed the Synthetic Minority Over-sampling Technique (SMOTE) to ensure even representation across all classes. This balancing prevents model bias towards more represented classes. Post-balancing, labels are encoded into a binary format, necessary for classification tasks in deep learning.

3.2.5 Partitioning Data for Model Training

The data is partitioned into training and testing sets, with a specific portion allocated for model validation. This split is crucial for evaluating the model's performance on unseen data, ensuring robustness and generalizability. The training set, which makes up the majority of the data, offers a wide range of examples for learning. On the other hand, the testing set is crucial for objectively assessing performance. The table below provides a clear breakdown of the number of image files generated and trained for each vulnerability (see Table 1).

Labels	No of image files generated
Block number dependency (BN)	284
Dangerous delegate call (DE)	436
Ether strict equality (SE)	312
Integer overflow (OF)	281
Re-entrancy (RE)	233
Timestamp dependency (TP)	260
Unchecked external call (UC)	211

Table 1: Vulnerability and image dataset trained

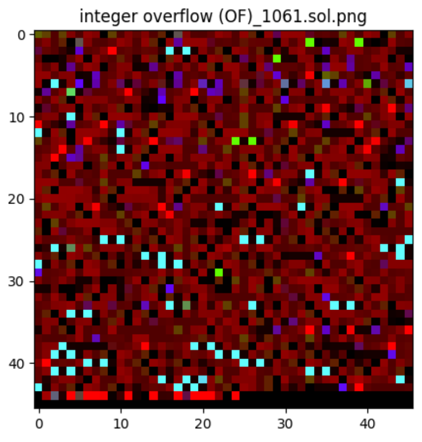


Figure 3 Example of Image date generated

3.3 Deep learning Model Training

The research employed a variety of deep learning models, chosen for their ability to recognise patterns and extract features from intricate datasets. The dataset includes features such as file_name, byte_code, and target. The models underwent training using a cross-validation strategy to avoid overfitting, while hyperparameter tuning was performed through grid search to improve performance. The effectiveness of the models was evaluated using various metrics including accuracy, precision, recall, and F1 score. The metrics offered a thorough grasp of the models' effectiveness in identifying vulnerabilities in smart contracts.

3.3.1 CNN Model Training and Evaluation:

The Convolutional Neural Network (CNN) utilised in this project has been specifically tailored to classify images that depict smart contract vulnerabilities. This CNN architecture is designed to effectively capture the intricate patterns in the image data, which are indicative of various smart contract vulnerabilities. The sequential layers of convolution, activation, pooling, and dropout, followed by fully connected layers, work together to achieve this goal. Batch normalisation and dropout are essential techniques for regularising the model, preventing overfitting, and promoting robust learning. This architecture is highly effective for the intricate task of image-based classification within the realm of smart contract security.

Convolutional Neural Network (CNN) Architecture Description

Here is a detailed architecture description based on the provided code:

Model Overview:

The neural network designed for this project features a specific architecture to handle image data effectively. It accepts input images of size 128x128 pixels with a depth of 3, representing RGB color channels, and dynamically adjusts the input shape based on the image data format (channels_first or channels_last). The architecture includes several convolutional layers: the first convolutional layer has 32 filters of size 2x2, uses 'same' padding, ReLU activation, followed by batch normalization and a MaxPooling layer with a pool size of 3x3, and includes a dropout of 0.25. The second convolutional layer is composed of a Conv2D layer with 64 filters of size 3x3, 'same' padding, ReLU activation, batch normalization, another Conv2D layer with 32 filters, and follows a similar pattern of batch normalization, MaxPooling, and dropout. The third convolutional layer mirrors the second, but with 64 and 128 filters in its Conv2D layers. After the convolutional layers, the network output is flattened into a single vector, which then passes through a fully connected dense layer with 1024 units, ReLU activation, batch normalization, and a final dropout of 0.25. The network culminates in an output layer, a dense layer with a number of units equal to the number of classes (n_classes), utilizing a softmax activation function suitable for multi-class classification.

Model Compilation
The model is compiled with the categorical cross entropy loss function and the stochastic gradient descent (SGD) optimizer. The primary metric that is used for comparison of the models is accuracy.

Model Training
The model is trained on the preprocessed dataset with a specified batch size (BATCHZ_SIZE =64) over 10 epochs. It utilizes both training and validation datasets (X_train, y_train and X_test, y_test) to learn and validate its performance.

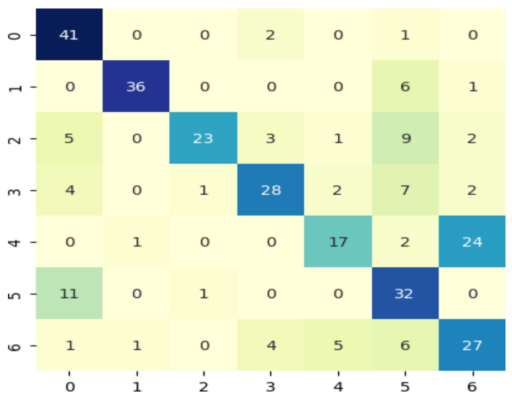
Model Performance

The CNN model's analysis presented a notable precision and recall, particularly for the detection of the vulnerability type represented by class 1 (refer Table 2). This indicates a robust ability to identify specific issues within smart contracts (Pouyan Momeni,2019). However, the moderate scores in class 4 and class 6 suggest a challenge in detecting certain vulnerabilities, possibly due to the model's limitations in capturing the complex patterns associated with these classes. The confusion matrix would likely show some degree of misclassification between these classes, signaling the need for more granular feature engineering or more training data to improve the model's discernment. The loss curve's progression points toward a stable learning process, but it also hints that the model might benefit from techniques to combat overfitting, such as dropout or regularization methods . This model's performance invites discussions on the architecture's intricacies, while for practitioners, it suggests utility with a scope for refinement.

Index	vulnerability
0	block number dependency (BN)
1	dangerous delegatecall (DE)

2	ether strict equality (SE)
3	integer overflow (OF)
4	reentrancy (RE)
5	timestamp dependency (TP)
6	unchecked external call (UC)

Table 2 Index number and vulnerability type



	0	1	2	3	4	5	6
0	41	0	0	2	0	1	0
1	0	36	0	0	0	6	1
2	5	0	23	3	1	9	2
3	4	0	1	28	2	7	2
4	0	1	0	0	17	2	24
5	11	0	1	0	0	32	0
6	1	1	0	4	5	6	27

Figure 5:Confusion Matric for CNN Model

3.3.2 Xception Model Training and Evaluation:

The Xception architecture used in this project is specifically designed for classifying smart contract vulnerabilities. This model tabiliz pre-trained weights from ImageNet and is specifically designed for input images that are 128x128 pixels in size and have 3 colour channels.

The Xception-based model has an advanced architecture that is enhanced by additional layers for pooling and tabilization . This makes it highly capable of classifying images that represent various vulnerabilities in smart contracts. Utilising a pre-trained base model expedites the learning process and has the potential to enhance the model's performance through transfer learning.

Model Configuration

The Xception model, a robust and efficient deep learning architecture, is used as the base model in this project. The configuration is designed to match the project's image dimensions and uses pre-trained weights from ImageNet. The top layer is excluded to provide flexibility for tabilization. The entire model is configured to be trainable, which means that the weights will be adjusted and refined during the training process. Extra layers are added to enhance the Xception base model. This includes a GlobalAveragePooling2D layer that helps reduce dimensionality by condensing feature maps into a single vector, which is beneficial for classification tasks. In order to address the issue of overfitting, a dropout layer is included in the model. This layer, with a rate of 0.5, randomly deactivates a portion of the input units during each training update. The architecture ends with a dense layer that has the same number of units as there are classes (7_classes). It uses a sigmoid activation function, which is suitable for multi-label classification tasks.

Model Compilation: The model is compiled using categorical cross-entropy as the loss function and Stochastic Gradient Descent (SGD) as the optimizer. Precision is the key factor in assessing performance.
Model Training: The model undergoes training on the dataset for 10 epochs, tabiliza a specified batch size. During this phase, both training and validation datasets are utilised to evaluate the performance on unseen data.

Model Performance:

The Xception model's execution yielded less accuracy overall compared to the CNN model. Its performance was particularly commendable in class 1, demonstrating the model's potential in recognizing certain vulnerabilities with high precision. The lower performance in other classes, as detailed by the confusion matrix, raises questions about the model's feature extraction capabilities and generalizability. The Xception architecture, known for its depthwise separable convolutions, might require a deeper dive into the hyperparameters to better tune the model for the dataset in question. The loss curve analysis for this model could provide insights into whether the training duration was adequate or if the model suffered from high bias or variance, which would be crucial for practical deployment.

Class	Precision	Recall	F1-Score	Support
0	0.67	0.77	0.72	43
1	0.98	1	0.99	44
2	0.79	0.75	0.77	44
3	0.81	0.67	0.73	43
4	0.46	0.36	0.41	44
5	0.76	0.7	0.73	44
6	0.41	0.55	0.47	44
Overall			0.69	306

Table 3 :Evaluation Metrics for Xception Model

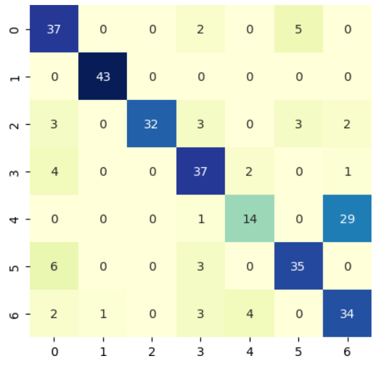


Figure 5:Confusion Matrix for Xception Model

3.1.1 EfficientNetB2 Architecture for Smart Contract Vulnerability Classification

The EfficientNetB2 model in this project is tailored for classifying images related to smart contract vulnerabilities. The model tabiliz the EfficientNetB2 architecture, which is renowned for its optimal combination of accuracy and efficiency. It has been specifically modified to handle input images of dimensions 128x128 pixels with 3 colour channels. EfficientNetB2 was chosen instead of higher versions like EfficientNetB7 because of its superior computational efficiency, which is crucial in situations where resources are limited. The architecture provides a practical balance, delivering impressive performance on smaller datasets commonly found in the niche field of smart contract vulnerabilities. In addition, EfficientNetB2 is a valuable option in situations that require quick prototyping and speedy iteration. It effectively addresses the challenge of limited computational resources while maintaining reliable vulnerability detection.

Model Configuration

The base model used in this project tabiliz the EfficientNetB2 architecture, which is well-known for its improved accuracy and efficiency among the EfficientNet family. The model is tabilizati with pre-trained ImageNet weights, excluding the top layer to enable additional tabilization. The input shape is specifically configured to match the project's image dimensions of 128x128 pixels with 3 colour channels. As part of the base model, we introduce a GlobalAveragePooling2D layer. This layer helps to reduce dimensionality by averaging spatial information. This simplifies the feature vector, making it easier to compute and reducing the risk of overfitting. Afterwards, a dense layer is included with a number of units that matches the number of classes (n_classes). It employs a softmax activation function, which is suitable for multi-class classification tasks.

Model Compilation: The model is compiled using categorical crossentropy as the loss function and the Adam optimizer. The Adam optimizer is highly regarded for its exceptional ability to handle sparse gradients and adapt learning rates. We have selected accuracy as the performance metric.

Model Training: The training is carried out over 10 epochs with a specified batch size. The model makes use of both training and validation datasets to monitor and adjust its performance based on feedback from the validation data.

The EfficientNetB2 architecture in this project is a deliberate choice for tackling the intricate task of image classification in the realm of smart contract vulnerabilities. This model's efficient design, along with its ability to leverage transfer learning from ImageNet weights, makes it a powerful and reliable tool for accurately classifying input images by extracting detailed features.
After completing the training process, the model is stored for future use, demonstrating its preparedness for deployment or additional assessment.

4 Design Specification and Implementation

The system's core comprises two distinct models: Flask Webapp, EfficientNetB2, each with unique operational paradigms tailored to identify patterns indicative of vulnerabilities within smart contract code. Based on the models that were trained, it was found that EfficientNet B2 was the most successful of the three models that were trained.

4.1 Webapp Design and implementation to test solidity files

The Flask web application is a popular choice due to its simplicity and effectiveness in handling web requests. The application's design focuses on processing Solidity smart contract files. It takes these files as input, extracts their bytecode, and then uses this bytecode for classification purposes.
This website provides a user-friendly platform for uploading Solidity files and receiving instant feedback on possible security risks. The website's backend compiles Solidity code and converts it into images. It is powered by the EfficientNetB2 algorithm, which has been fine-tuned for smart contract code analysis. This model was chosen for its computational efficiency and its demonstrated accuracy in classification tasks in limited environments.

The tabilizature of EfficientNetB2 is well-suited for processing the intricate patterns present in Solidity smart contracts, thanks to its balanced scaling approach. The assessment of the code is fast and dependable, as it analyses the complex structures and flow within the contracts to identify vulnerabilities.

0.1 Design Specification for a Flask-Based Web Application for Solidity File Analysis

4.2.1 Setup and Libraries:

Essential libraries like solcx, numpy, PIL, and efficientnet.keras are used for tasks ranging from Solidity compilation, image processing, to running neural network models. Flask serves as the backbone of the web application.

4.2.2 Flask Application Framework:

The application establishes routes for home and classification pages and the main classification endpoint, facilitating user interaction with the system for vulnerability analysis.

The home page in the application serves as the entry point for users, providing a user-friendly interface to engage with the system. It typically includes instructions, background information, or an overview of the application's purpose, setting the stage for a seamless user experience.

The classification page is where users can interact directly with the vulnerability analysis feature. Here, users can upload smart contract code, initiate analysis, and receive feedback. This page is crucial as it operationalizes the application's core functionality, transforming it from a theoretical tool into a practical instrument for real-world use.

The main classification endpoint is the back-end component that processes the user requests. It receives the smart contract data, runs the classification using the EfficientNetB2 model, and returns the analysis results. This endpoint is critical for the actual computation and delivery of the vulnerability assessment, enabling the application to serve its primary function of smart contract vulnerability detection.

4.2.3 Solidity File Upload and Processing :

The application allows users to upload Solidity files. These files, written in the Solidity programming language, are the backbone of Ethereum-based smart contracts. Upon upload, the system conducts preliminary checks to ensure file integrity and compatibility.

4.2.4 Bytecode Extraction:

Once a Solidity file is uploaded, the system selects the correct Solidity compiler version and compiles it into bytecode using. This transformation is critical as bytecode encapsulates the operational logic of the contract, which is essential for subsequent analysis. The compilation process is designed to be efficient and error-resistant, ensuring accurate bytecode generation.

4.2.5 Image Generation from Bytecode:

Ethereum Virtual Machine (EVM) bytecode is converted into RGB images for deep earning analysis. The process involves normalizing the bytecode for consistency, then reshaping and converting it into an image format. Each byte of the bytecode is represented as a pixel in the RGB image. These images are saved and used for further analysis, providing a visual method to study and interpret blockchain smart contracts.

4.2.6 EfficientNet B2 Model Integration:

The choice of EfficientNetB2 was mainly driven by its architecture, which is a part of the larger EfficientNet family renowned for its structured and efficient approach to scaling model size. This architecture achieves a perfect balance, improving accuracy without adding too much computational burden. This project benefits greatly from the model's ability to process input images of 128x128 pixels with three colour channels. This seamlessly aligns with the nature of images derived from smart contract bytecode, allowing for the extraction of detailed features that are crucial for identifying vulnerabilities.

Once the training was complete, the EfficientNetB2 model was exported as an H5 file, ensuring that its architecture, weights, and configuration were preserved. This step ensures that the model is prepared for efficient deployment and reuse, while maintaining its performance. Incorporating this model into the web application was a crucial stage. By incorporating EfficientNetB2, we were able to achieve real-time analysis and classification of smart contract images uploaded by users. This integration successfully connects the world of theoretical modelling with practical applications.

6 Evaluation

6.1 Choice of Evaluation Metrics:

This research used a number of deep learning models, each chosen for their ability to   tabiliza patterns and extract features from the dataset. The effectiveness of these models were evaluated using different metrics: accuracy, precision, recall, and F1 score. These metrics were selected because they offer a thorough insight into the performance of the models in identifying vulnerabilities in smart contracts. The accuracy metric evaluates the model's ability to correctly classify vulnerabilities. Understanding the model's ability to identify true positives and   tabiliz false positives is crucial when it comes to precision and recall. The F1 score is a balanced measure of the model's precision and recall capabilities, calculated as the harmonic mean of precision and recall. These metrics, when combined, provide a comprehensive assessment of the models' performance.

The key evaluation metric for measuring the effectiveness of the end-user system in this research was the number of vulnerabilities it could detect within smart contracts. This quantitative measure directly reflects the website's ability to identify and flag potential security issues, providing a clear benchmark for comparing its performance with previous studies and tools in the field. The more vulnerabilities the site can reliably detect, the more valuable it becomes as a tool for developers and auditors seeking to ensure the security and integrity of smart contract code.

6.1.1 Experiments to Test the Efficiency of the Web Application:

To validate the efficiency of the web application that employs the EfficientNet B2 algorithm, several experiments were conducted. The models, including EfficientNet B2, were trained using a cross-validation strategy to prevent overfitting, with hyperparameter tuning conducted via a grid search to enhance performance. This approach ensured that the models were well-optimized for detecting various types of vulnerabilities in smart contracts. The web application was then tested by processing real-world smart contract codes. These codes were uploaded to the application, compiled into bytecode, and analyzed by the EfficientNet B2 model. The model's performance was evaluated based on its ability to accurately classify and identify potential vulnerabilities in these contract.

6.2 Experiment Analysis 3: EfficientNet B2

EfficientNet B2 has proven to be the top-performing model, achieving the highest accuracy and demonstrating an impressive balance across classes. The model's effectiveness in differentiating between different vulnerability types indicates a superior ability to extract features and learn. Practitioners can clearly see that this model is ready for deployment, although it is advisable to continuously monitor and update the model. The confusion matrix for EfficientNet B2 suggests a finely-tuned model that holds promise for real-world applications. The model's impressive performance academically can inspire further research into scalable architectures such as EfficientNet. The loss curve illustrates an ideal training process, serving as a reference point for future experimen's in related fields. Figure 6 demonstrates the convergence and tabilization of the training and loss curve, resulting in the optimal scenario for the EffecientNet B2 algorithm.

Users can easily observe how the model categorises various aspects of the uploaded smart contracts, gaining valuable insights into its decision-making process. Furthermore, it acts as an invaluable educational resource, aiding users in comprehending the specific vulnerabilities that the model is highly sensitive to. The loss curve (refer figure 6 and 8), showcasing an ideal training process, establishes a benchmark for future deep learning projects in the cybersecurity field. The confusion matrix (refer figure 7) serves as a quantitative illustration of EfficientNet B2's learning efficiency and can be used as a reference for developing similar models for other applications. The deployment of EfficientNet B2 through a web interface represents a noteworthy advancement in translating academic research into practical applications. This resource effectively connects theoretical models with practical tools, offering the banking sector a valuable asset to strengthen the security of Solidity smart contracts. Here is the table(Table 4) that provides the values of the metrics used to evaluate the efficiency of the EfficientNet B2 algorithm.

Table 4: Evaluation Metrics for EfficientNet B2

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.76	0.8	0.78	44
1	0.98	1	0.99	44
2	0.93	0.95	0.94	43
3	0.82	0.84	0.83	44
4	0.66	0.48	0.55	44
5	0.88	0.67	0.76	43
6	0.57	0.8	0.67	44
Overall			0.79	306

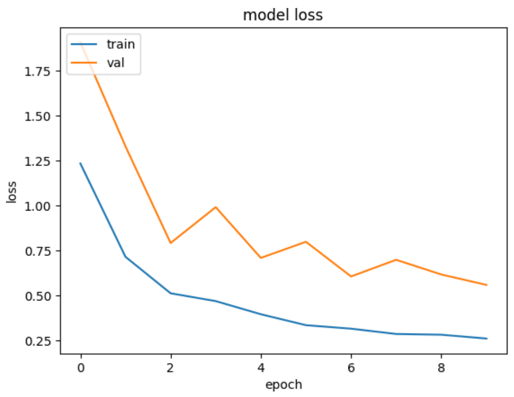


Figure 6: Accuracy over Epochs

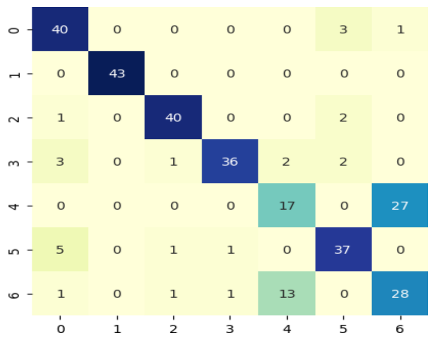


Figure 7: Confusion Matrix for EfficientNet B2

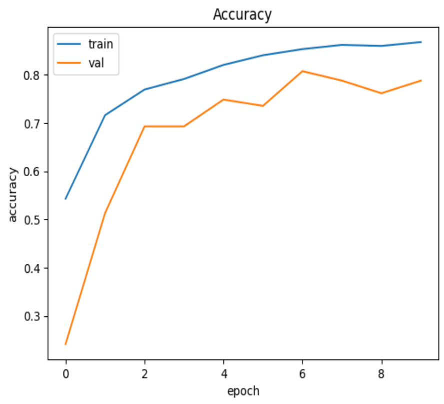


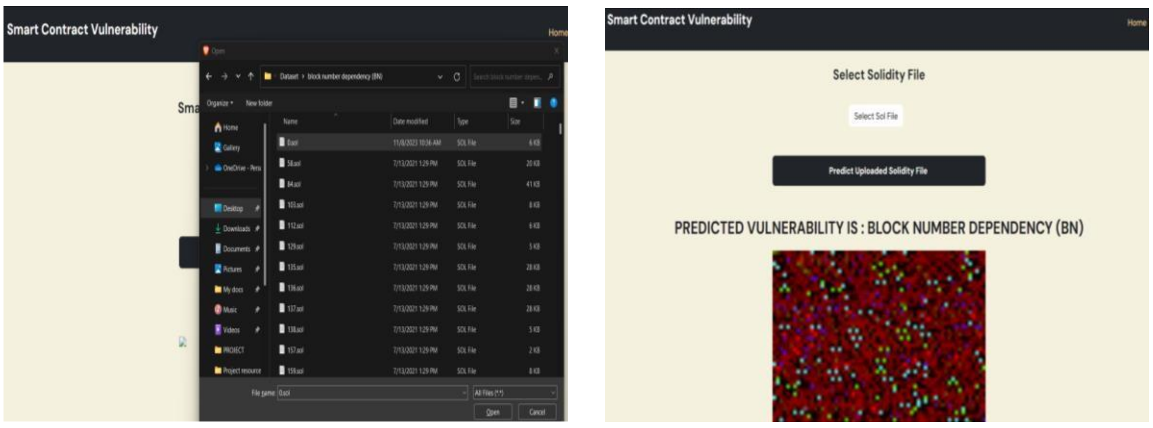
Figure 8: Training and validation losses over epochs

6.3 WEBAPP:

In the web interface, users, predominantly developers and security auditors, can upload Solidity smart contract files for evaluation. The process is straightforward and user-friendly: once a file is uploaded, EfficientNet B2 processes the file, analyzing the contract's bytecode to identify potential security vulnerabilities. This analysis leverages the model's trained capacity to distinguish between multiple vulnerability types, providing a nuanced assessment of the code. A total of ten tests, (shown in Table 5) were conducted, and out of these, only one test detected the vulnerability incorrectly (Refer Figure 10).

Test Case Number:	Vulnerability Name	Pass/Fail
1	Block Number Dependency	Pass
2	Dangerous Delegate Call	Pass
3	Ether Strict Equality	Pass
4	Integer Overflow	Pass
5	Block Number Deependency	Fail
6	Reentrancy	Pass
7	Timestamp Depenedency	Pass
8	Unchecked external Call	Pass
9	Reentarncy	Pass
10	Ether Strict Equality	Pass

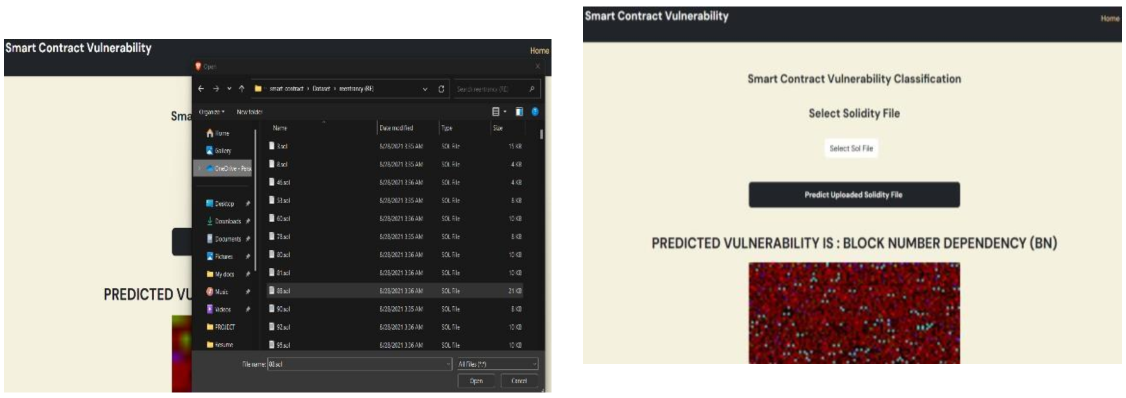
Table 5: Website test Pass/Fail Cases



```
Name       Date modified         Type      Size
0.sol      11/8/2023 10:36 AM    SOL File   6 KB
58.sol     7/13/2021 1:29 PM     SOL File  20 KB
84.sol     7/13/2021 1:29 PM     SOL File  41 KB
103.sol    7/13/2021 1:29 PM     SOL File   8 KB
112.sol    7/13/2021 1:29 PM     SOL File   6 KB
129.sol    7/13/2021 1:29 PM     SOL File   5 KB
135.sol    7/13/2021 1:29 PM     SOL File  28 KB
136.sol    7/13/2021 1:29 PM     SOL File  28 KB
137.sol    7/13/2021 1:29 PM     SOL File  28 KB
138.sol    7/13/2021 1:29 PM     SOL File   5 KB
157.sol    7/13/2021 1:29 PM     SOL File   2 KB
159.sol    7/13/2021 1:29 PM     SOL File   8 KB
File name: 0.sol    All Files (*.*)
```

PREDICTED VULNERABILITY IS : BLOCK NUMBER DEPENDENCY (BN)

Fig 9: Vulnerability Prediction done using the Webapp (Correct Prediction)



```
Name      Date modified        Type      Size
3.sol     5/28/2021 3:35 AM    SOL File  15 KB
8.sol     5/28/2021 3:35 AM    SOL File   4 KB
45.sol    5/28/2021 3:36 AM    SOL File   4 KB
53.sol    5/28/2021 3:35 AM    SOL File   8 KB
60.sol    5/28/2021 3:36 AM    SOL File  10 KB
78.sol    5/28/2021 3:35 AM    SOL File   8 KB
80.sol    5/28/2021 3:36 AM    SOL File  10 KB
81.sol    5/28/2021 3:36 AM    SOL File  10 KB
88.sol    5/28/2021 3:36 AM    SOL File  21 KB
90.sol    5/28/2021 3:35 AM    SOL File   8 KB
92.sol    5/28/2021 3:36 AM    SOL File  10 KB
95.sol    5/28/2021 3:36 AM    SOL File  10 KB
File name: 88.sol    All Files (*.*)
```

PREDICTED VU

PREDICTED VULNERABILITY IS : BLOCK NUMBER DEPENDENCY (BN)

Fig 10: Vulnerability Prediction done using the Webapp(Incorrect Prediction)

6.4 Discussion:

EfficientNet B2 emerged as a standout in this research, showcasing the highest overall accuracy at 79% among the models tested. This model achieved high precision and recall rates across different classes of vulnerabilities, indicating a robust capacity for feature extraction and generalization. For instance, in one class, it attained a precision rate of 98% and a recall rate of 100%, culminating in an F1-score of 99%. Such metrics underscore the model's adeptness in capturing and classifying the nuances of smart contract vulnerabilities. Notably, its precision and recall were consistently high, with scores such as 93% precision and 95% recall in another class, demonstrating its reliability across various types of vulnerabilities. The Xception model, while presenting a slight decrease in overall accuracy at 69%, maintained commendable precision, particularly in class 1, where it achieved a precision rate of 98% and a recall of 100%. Although the performance in other classes was not as high as EfficientNet B2, with precision and recall rates at 46% and 36% in one of the lower-performing classes, it still demonstrated potential in recognizing definitive patterns associated with certain vulnerabilities. CNN on the other hand exhibited the lowest accuracy value with 67%.

Comparing these models with literature references highlights their strengths and potential areas for improvement. Pouyan Momeni’s work reported an accuracy of 95% with an F1-score of 79%, indicating that while the models in this research did not surpass that accuracy, they offer competitive performance with the benefit of enhanced feature extraction capabilities. Mezina’s research on the other hand discusses combined binary and multiclass classification approaches for vulnerability detection, which could complement the deep learning models employed in this research by providing a broader scope in vulnerability identification. Unlike static code analyzers, deep learning models used in this research offer a more effective solution by detecting a wider range of vulnerabilities and reducing processing time. They provide the benefit of analysing intricate patterns within smart contracts with greater efficiency and effectiveness.

The study's utilisation of EfficientNet B2 via a web interface represents a notable breakthrough in bridging the gap between academic research and practical implementation. It offers a valuable tool for developers and security auditors to evaluate smart contracts for vulnerabilities with increased accuracy and efficiency. According to the research, no single model was found to be superior in all aspects. However, the CNN, EfficientNet B2, and Xception models each have their own unique strengths. EfficientNet B2 is particularly notable for its well-rounded performance across different vulnerability types, making it a promising choice for practitioners in search of dependable tools for smart contract security analysis. As deep learning architectures continue to evolve, these models are becoming more advanced, providing stronger protection against the vulnerabilities that can compromise the security of smart contracts on the blockchain.

7. Conclusion and Future Work

This study has investigated the utilisation of deep learning techniques for identifying vulnerabilities in smart contracts, which is an essential undertaking. The study employed three different models: CNN, Xception, and EfficientNet B2. Each model showcased unique strengths and characteristics within this specific context. The research conducted a comprehensive comparative analysis, which yielded valuable insights into the performance of these models. In the end, the study revealed that the EfficientNet B2 model showcased exceptional performance, boasting an impressive accuracy rate of 79%. The achievement serves as evidence of the effectiveness in detecting vulnerabilities in smart contracts, as demonstrated by the high precision, recall, and F1-scores achieved across different classes.

The study's findings confirm that the CNN, EfficientNet B2, and Xception algorithms, especially EfficientNet B2, have the potential to surpass current vulnerability detection methods. The successful implementation of these algorithms in a practical web application framework represents a crucial advancement in enhancing the security of smart contracts in the blockchain ecosystem.

This research has far-reaching implications, as it has effectively confirmed the initial hypothesis that deep learning models can serve as powerful tools for identifying potential vulnerabilities in smart contracts. This fulfilled a crucial requirement in the world of blockchain technology, where the trustworthiness and credibility of decentralised systems depend heavily on the security of smart contracts. There have been growing concerns about the security of smart contracts due to the increasing use of blockchain applications across various industries. The research presented in this study offers a hopeful solution to tackle this complex issue.

This research presents numerous possibilities for advancement and further exploration. Firstly, there is a great opportunity for data enrichment by expanding the dataset to incorporate a broader range of vulnerabilities and different smart contract architectures. This expansion has the potential to enhance the model's capability in identifying a wider range of security concerns, thus further bolstering the security of blockchain-based systems.

It is crucial to conduct additional research on model optimisation for future investigations. Exploring different model architectures and optimising hyperparameters can greatly enhance the model's predictive abilities and improve detection rates. These optimisations can greatly improve the accuracy and reliability of vulnerability detection mechanisms.

References

Momeni, P., Wang, Y. and Samavi, R., 2019, August. deep learning model for smart contracts security analysis. In 2019 17th International Conference on Privacy, Security and Trust (PST) (pp. 1-6). IEEE.

Mezina, A. and Burget, R., 2022, October. Obfuscated malware detection using dilated convolutional network. In 2022 14th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT) (pp. 110-115). IEEE.

Sklaroff, J.M., 2017. Smart contracts and the cost of inflexibility. U. Pa. L. Rev., 166, p.263.

Narayana, K.L. and Sathiyamurthy, K., 2021. Automation and smart materials in detecting smart contracts vulnerabilities in blockchain using deep learning. Materials Today: Proceedings.

Deng, W., Wei, H., Huang, T., Cao, C., Peng, Y. and Hu, X., 2023. Smart contract vulnerability detection based on deep learning and multimodal decision fusion. Sensors, 23(16), p.7246.

Hwang, S.J., Choi, S.H., Shin, J. and Choi, Y.H., 2022. CodeNet: Code-targeted convolutional neural network architecture for smart contract vulnerability detection. IEEE Access, 10, pp.32595-32607.

Sosu, R.N.A., Chen, J., Brown-Acquaye, W., Owusu, E. and Boahen, E., 2022. A Vulnerability Detection Approach for Automated Smart Contract Using Enhanced deep learning Techniques.

Owasp Smart Contract top OWASP Smart Contract Top 10 | OWASP Foundation. Available at: https://owasp.org/www-project-smart-contract-top-10/ (Accessed: 04 January 2024).

Ganesh, M., Pednekar, P., Prabhuswamy, P., Nair, D.S., Park, Y. and Jeon, H., 2017, July. CNN-based android malware detection. In 2017 international conference on software security and assurance (ICSSA) (pp. 60-65). IEEE.

Tan, M. and Le, Q., 2019, May. Efficientnet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning (pp. 6105-6114). PMLR.

Kuo, W.C. and Lin, Y.P., 2019. Malware detection method based on CNN. In New Trends in Computer Technologies and Applications: 23rd International Computer Symposium, ICS 2018, Yunlin, Taiwan, ember 20–22, 2018, Revised Selected Papers 23 (pp. 608-617). Springer Singapore.

Zhang, J., Qin, Z., Yin, H., Ou, L. and Zhang, K., 2019. A feature-hybrid malware variants detection using CNN based opcode embedding and BPNN based API embedding. Computers & Security, 84, pp.376-39

Ravi, V. and Chaganti, R., 2023. EfficientNet deep learning meta-classifier approach for image-based android malware detection. Multimedia Tools and Applications, 82(16), pp.24891-24917.

Ibrahim, N.M., Gabr, D.G., Rahman, A., Musleh, D., AlKhulaifi, D. and AlKharraa, M., 2023. Transfer Learning Approach to Seed Taxonomy: A Wild Plant Case Study. Big Data and Cognitive Computing, 7(3), p.128.

Yadav, P., Menon, N., Ravi, V., Vishvanathan, S. and Pham, T.D., 2022. EfficientNet convolutional neural networks-based Android malware detection. Computers & Security, 115, p.102622.

Taherdoost, H., 2023. Smart Contracts in Blockchain Technology: A Critical Review. Information, 14(2), p.117.

Wang, W., Song, J., Xu, G., Li, Y., Wang, H. and Su, C., 2020. Contractward: Automated vulnerability detection models for ethereum smart contracts. IEEE Transactions on Network Science and Engineering, 8(2), pp.1133-1144.

Sendner, C., Chen, H., Fereidooni, H., Petzi, L., König, J., Stang, J., Dmitrienko, A., Sadeghi, A.R. and Koushanfar, F., 2023. Smarter Contracts: Detecting Vulnerabilities in Smart Contracts with Deep Transfer Learning. In NDSS.

Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1251-1258).

Xu, Y., Hu, G., You, L. and Cao, C., 2021. A novel deep learning-based analysis model for smart contract vulnerability. Security and Communication Networks, 2021, pp.1-12.

Jiang, B., Liu, Y. and Chan, W.K., 2018, September. Contractfuzzer: Fuzzing smart contracts for vulnerability detection. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (pp. 259-269).

Chen, J., Xia, X., Lo, D., Grundy, J., Luo, X. and Chen, T., 2021. Defectchecker: Automated smart contract defect detection by analyzing evm bytecode. IEEE Transactions on Software Engineering, 48(7), pp.2189-2207.

Brent, L., Grech, N., Lagouvardos, S., Scholz, B. and Smaragdakis, Y., 2020, June. Ethainter: a smart contract security analyzer for composite vulnerabilities. In Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation (pp. 454-469).

Zhao, Z., Li, Z., Yu, J., Zhang, F., Xie, X., Xu, H. and Chen, B., 2023. CMD: Co-analyzed IoT Malware Detection and Forensics via Network and Hardware Domains. IEEE Transactions on Mobile Computing.

Singh, A., Parizi, R.M., Zhang, Q., Choo, K.K.R. and Dehghantanha, A., 2020. Blockchain smart contracts formalization: Approaches and challenges to address vulnerabilities. Computers & Security, 88, p.101654.

Fu, Y., Li, C., Yu, F.R., Luan, T.H., Zhao, P. and Liu, S., 2022. A survey of blockchain and intelligent networking for the metaverse. IEEE Internet of Things.