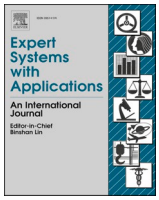
Expert Systems With Applications 268 (2025) 126353

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



Review

Machine learning methods for detecting smart contracts vulnerabilities within Ethereum blockchain – A review

Joao Crisostomo [a,*], Fernando Bacao [a], Victor Lobo [a,b]

[a] Nova IMS Lisbon Portugal
[b] CINAV Portuguese Naval Academy Almada Portugal

ARTICLE INFO

Keywords:
Review
Smart Contract
Vulnerabilities
Machine Learning
Deep Learning

ABSTRACT

This paper presents a comprehensive exploration of the intersection between machine learning and smart contract vulnerabilities on the Ethereum blockchain. Introduced by Vitalik Buterin in 2015, Ethereum stands as a prominent blockchain network, necessitating innovative approaches to secure smart contracts against vulnerabilities and potential attacks. This research follows PRISMA guidelines, posing three fundamental questions and conducting a meticulous literature review. The study categorises machine learning applications into seven distinct groups, analysing their taxonomy, feature types, and engineering methods. The findings indicate a dynamic landscape characterised by a noticeable trend towards increased complexity. This complexity is evident not only in the integration of machine learning frameworks that combine different architectures of deep learning models, such as Convolutional Neural Networks (CNN), Graph Neural Networks (GNN), or Recurrent Neural Networks (RNN), but also in the incorporation of various types of data related to smart contracts (SCs). The discussion dissects the advantages, limitations, and future directions in securing smart contracts using machine learning. The paper concludes by emphasising the evolving role of machine learning in strengthening the Ethereum blockchain, fostering trust, and enhancing security in decentralised systems.

1. Introduction

Ethereum, introduced by Vitalik Buterin in 2015 (Buterin, 2022), stands as the second-largest blockchain network by market cap, surpassed only by Bitcoin (as of December 8, 2023, according to CoinMarketCap) (Coin, 2024). The Ethereum network comprises two main account types: Ethereum Owned Account (EOA) and Smart Contract Account (ERC-20) (Buterin, 2022; Zhang et al., 2022). The latter represents a self-executable code that can be invoked by another Smart Contract or EOA, using a Turing-complete programming language (Dannen, 2017).

The cryptographic and security elements embedded in smart contracts (SCs) play a pivotal role in facilitating transparent and globally accessible transactions, eliminating the need for third-party validators or intermediaries (Atzei et al., 2017). This technological innovation has ushered in a new era in blockchain technology, known as Blockchain 2.0 (Kehrli, 2018). This era empowers users to develop sophisticated applications on blockchain networks, enabling decentralised, self-executing agreements across diverse sectors such as the Internet of Things (Lo et al., 2019), Healthcare (Merlo et al., 2023), Commercial services (Anoop et al., 2022), and secure data exchange (Wang and Guan, 2023).

The rise of smart contracts, however, brings with it a set of challenges, with security emerging as the main concern (Atzei et al., 2017). As individuals amass substantial currency holdings within these contracts, the risk of code vulnerabilities escalates (Jain and Tripathi, 2023). This susceptibility becomes a focal point for attackers and manipulators to exploit these weaknesses, resulting in a series of attacks on smart contracts in recent years (Hacks, 2024). These attacks are summarised in Table 1 (Hacks, 2024).

Motivated by the financial impact highlighted in Table 1, which underscores the critical need to address these vulnerabilities, the contribution of this survey lies in applying machine learning methods to detect smart contract vulnerabilities. This survey focuses on examining how machine learning methodologies can strengthen smart contracts against potential exploitation and hacking, thereby mitigating financial losses and promoting the adoption of this technology across a broader set of applications within society. Initially, these vulnerabilities were identified using expert knowledge, predefined patterns or fuzzy methods such as Mythrill, Osiris, Oyente, Securify or SmartCheck (Kushwaha

* Corresponding author.
E-mail addresses: jdiogo.rcosta@gmail.com (J. Crisostomo), bacao@novaims.unl.pt (F. Bacao).

https://doi.org/10.1016/j.eswa.2024.126353
Received 1 March 2024; Received in revised form 2 November 2024; Accepted 28 December 2024
Available online 3 January 2025

Table 1
Main Ethereum blockchain attacks in 2023, in (Hacks, 2024).

Author	Date	Technique	Lost amount
Orbit Bridge	31 Dec 2023	Signature Exploit	$81,7M
KyberSwap Elastic	22 Nov. 2023	Flashloan Swap Logic Exploit	$48,0M
Heco Bridge	22 Nov. 2023	Private Key Compromised	$86,6M
dYdX	18 Nov. 2023	Price Manipulation Attack	$9,0M
Kronos Research	18 Nov. 2023	API Key unauthorised access	$26,0M
Poloniex	10 Nov. 2023	Private Key Compromised	$126,0M
HTX	25 Sep 2023	Private Key Compromised	$8,0M
Mixin Network	23 Sep. 2023	Database	$200,0M
Zunami Protocol	13 Aug 2023	Price Manipulation Attack	$2,1M
Earning.Farm	9 Aug 2023	Reentrancy	$0,5M
Steadefi	8 Aug 2023	Deployer Wallet Compromised	$1,2M
Curve	30 Jul. 2023	Vyper Compiler Bug	$61,7M
Sturdy	12 Jun. 2023	Flashloan Reentrancy Attack	$0,8M
Yearn	13 Apr 2023	Flashloan Misconfiguration Exploit	$11,5M
Euler Finance	13 Mar. 2023	Flashloan Donate Function Logic Exploit	$197,0M

et al., 2022). In the majority of the reviewed articles, these tools served as labelling tools to then train the machine learning models Table 2.

By bridging the gap between blockchain and machine learning domains, this survey contributes to the evolving discourse on securing the decentralised future envisioned by smart contracts, laying the groundwork for future research and advancements in this domain.

The subsequent sections are structured as follows: Section 2 outlines the Research Methodology, detailing the survey's approach. In Section 3, the Review Applications section is categorised into seven distinct groups, highlighting the various machine learning methods employed to tackle smart contract vulnerabilities. Finally, Section 4, the Discussion, provides a comprehensive synthesis of the research findings, drawing insights from the preceding subsections. Our conclusions are presented in Section 5.

2. Research methodology

As stated in the introduction, the main objective of this paper was to review the role of machine learning in addressing the smart contracts vulnerability detection for Ethereum blockchain. Therefore, this research was conducted, following the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) guidelines (Page et al., 2021), reviewing the present state of knowledge of applications of machine learning methods in Ethereum Blockchain.

During this investigation, four questions were formulated and subsequently explored within this article. The research questions are as follows:

RQ1. What is the taxonomy of machine learning methods used for smart contract vulnerability detection in Ethereum Blockchain and what are the strengths and limitations of these methods in identifying vulnerabilities?

RQ2. What feature types and feature engineering methods are employed to address smart contract vulnerabilities in the Ethereum Blockchain??

RQ3. What are the key challenges and the next steps within smart contract vulnerabilities for the Ethereum blockchain?

RQ4. What are the trade-offs between model complexity and performance?

Based on the research questions, the review was undertaken to address the following specific goals:

- To review machine learning methods performance and effectiveness in detecting smart contract vulnerabilities in Ethereum blockchain.
- To aggregate the Machine Learning trends and performances achieved in a single source of information
- To structure the work that has been done and define clear next steps regarding all the spread expertise.

We start our investigation by conducting an extensive literature search across three scientific abstract databases (Scopus, Web of Science, and IEEE). To collect the required articles, searching only journal articles and conference proceedings in english with the desired keyword in the Title, Abstract, or declared Keywords, the following search string was used:

(TITLE-ABS-KEY ((“Machine Learning” AND “Smart Contract” AND “Vulnerabilities” AND “Ethereum”))The search string was modified to meet the requirements and limitations of each selected search engine. The search was conducted between the 18th and 21th of June 2024. Table 1 shows the number of results obtained from each database.

With these 136 collected articles we checked for duplicates, ending with a total of 78 unique research articles. Then we filter the articles published before 1st of January of 2024 and perform a full-text analysis on the remaining 76 studies, resulting in a selection of 36 relevant studies for our research. It is important to highlight that the majority of the excluded papers were addressing IOT applications using the Ethereum blockchain or security concerns not directly relevant to the intersection of smart contract vulnerabilities and machine learning. We then added 15 articles identified via reference followup from those 36 relevant studies ending with a total of 51 articles that were included in this review. The full process of paper selection is presented in Fig. 1 and the detailed analysis is available in (Website, 2024).

Table 2
Database with the total number of search articles results.

Database Name	URL	No. Total Results
Scopus	scopus.com	65
Web of Science	webofknowledge.com	34
IEEE	ieee.com	37
Total		136

2.1. Similar reviewed papers

Several comprehensive reviews delve into the core aspects of Blockchain Technology and Machine Learning classification, and their collaborative role in identifying anomalies through ensemble methods.

Kushwaha et. al. (Kushwaha et al., 2022) conduct a systematic review of security vulnerabilities in the Ethereum blockchain. They begin by presenting the Ethereum blockchain architecture and the structure of a blockchain node. The paper then explores vulnerabilities in smart contracts, detailing their main root causes and the symbolic tools used for mitigation. However, the study does not specifically address machine learning methods to overcome these limitations.

Wenhan Hou et al. (Hou et al., 2021) conducted a comprehensive survey on Blockchain data analysis, with a specific focus on Bitcoin and Ethereum. The authors organised the research into four main categories: Security (comprising Network, Account, and Public Security), Privacy (divided into Privacy Threats and Privacy Protection), Performance (encompassing Influence Factors, Transaction Confirmation, and Transaction Congestion), and Prediction of Price.

In (Hisham et al., 2022), a survey is presented with three main

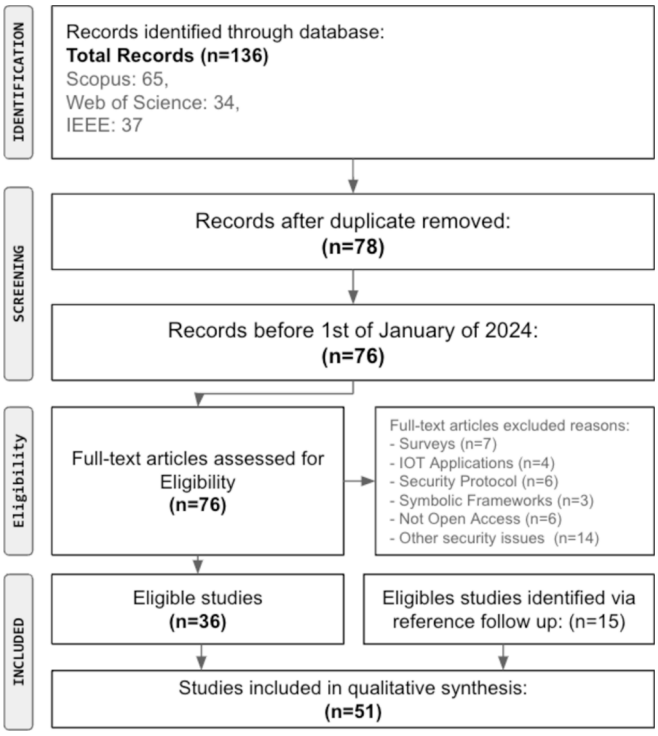


Fig. 1. PRISMA flow diagram explaining the eligibility criteria for the articles extracted from the scientific databases.

sections: exploration of Blockchain principles, an overview of ensemble learning classification, and the development of an ensemble learning approach for detecting anomalies in blockchain networks. The paper categorises anomalies into Security Aspect (threats from hackers), Information Processing Aspects (focusing on data dissemination processes), and Smart Devices Aspect (IoT devices connected using Blockchain Networks). It addresses not only machine learning methods to detect smart contract vulnerabilities but also other anomaly detection issues within Ethereum and Bitcoin blockchain networks.

Jiang et. al. (Jiang et al., xxxx) conducted a thorough examination of machine learning methods addressing smart contract vulnerabilities, offering detailed descriptions of each employed technique. However, notable research works like VulHunter by Li et al. (Li et al., 2023) and MANDO-HGT (Nguyen et al, xxxx) were overlooked. Additionally, the study lacked a structured presentation and comprehensive comparison of each machine learning approach group, missing an opportunity for a detailed overview.

As we synthesise these insights, it becomes apparent that while there are extensive reviews on Machine Learning applications for Blockchain in general or security evaluations of smart contract vulnerabilities, a specific emphasis on Machine Learning methods to detect and prevent smart contracts Vulnerabilities on the Ethereum blockchain, including a detailed comparison among the research works, is conspicuously lacking. This study addresses this gap, aiming to offer a dedicated exploration of this crucial intersection, paving the way for future research and advancements in the field of Smart Contract vulnerability detection using machine learning methods.

3. Review of applications

To provide a clear, structured, and concise overview of machine learning approaches and trends used in detecting vulnerabilities in smart contracts, we categorised the studies based on the type of machine learning solution presented in each. In cases where multiple machine learning approaches were employed, we selected the one demonstrating the best performance. Consequently, we divided the studies into different categories, including Supervised Learning, such as Decision Trees (DT) and Support Vector Machines (SVM), Ensemble Learning (Ex: Random Forest – RF) and Deep Learning (DL) methods (Mahesh, 2020). Within deep learning, we further categorised them into Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), Graph Neural Networks (GNN), following the classification of Deep Neural Network (DNN) architectures presented in (Shrestha and Mahmood, 2019). We also created a categorization of customised deep learning architectures (Customised DL Architectures).. Furthermore, given the heightened focus on Large Language Models (LLMs) over the past year, we opted to create a dedicated category for smart contract vulnerabilities addressed by LLMs. Fig. 2 illustrates the categorization of the presented machine learning methods, representing the criteria used to classify the articles in this review.

To maintain coherence across the reviewed articles and the vulnerabilities discussed, this review adopts the vulnerability categorization outlined in (Zaazaa and Bakkali, 2023). In terms of the primary vulnerability types explored in each article, Reentrancy (SW-107) emerges as the most extensively studied, closely followed by block timestamp/number manipulation (SW-116) and Integer Over/Underflow (SW-101). Additionally, the machine learning category – Graph Neural Networks (GNNs) exhibits the highest level of diversity, while Supervised Learning resides at the opposite end of the spectrum, Fig. 3.

Furthermore, due to the imbalanced nature of datasets used in vulnerability detection, we focus on the F1-score metric as it better reflects the true performance of the model. However, some studies did not provide this metric, nor could it be derived from the results presented. In these cases, we report the available performance metrics in the following order of preference: F1-Score, Precision and Recall, and finally, ROC-AUC. Additionally, only the highest-reported performance metric for each research article is included in this review.

3.1. Supervised Methods

Within the supervised methods we have one of the first articles to tackle the detection of smart contract vulnerabilities using machine learning. Momeni et. al. (Momeni et al., 2019) elaborates a study that utilises security vulnerabilities identified by static code analyzers, namely Mythril and Slither, to assess four machine learning models—SVM (Support Vector Machine), NN (Neural Networks), Decision trees, and RF (Random Forest)—as binary classifiers. The framework achieves an average accuracy of 95 %, predicting 16 major vulnerabilities. The proposed machine learning approach significantly outpaces static code analyzers, being approximately 22,800 times faster. This highlights its potential as a more efficient method for timely vulnerability detection in smart contracts, especially during the rapid development process.

SoliAudit (Liao et al., xxxx), uses a Logistic Regression trained with features generated from n-gram combined with term frequency–inverse document frequency (TF-IDF) and a CNN trained with features extracted from Word2Vec, being the first approach the one with better performance, achieving a 90 % accuracy in identifying 13 vulnerabilities. The author also developed a dynamic fuzzer component that operates by generating a fuzzer contract and deploying it alongside the target contract on the Ethereum blockchain. The fuzzer dynamically varies parameters such as gas, ether, and function calls to simulate different transaction scenarios. During the fuzzing process, the dynamic fuzzer actively interacts with the target contract, inserting events and analysing anomalies in transaction data. It employs anomaly detection techniques, calculating the Minkowski distance for each transaction and utilising median-absolute-deviation for outlier analysis. This approach helps identify potential weaknesses, such as reentrancy and arithmetic overflow problems, without relying on predefined patterns or expert knowledge.

Starting from the Abstract Syntax Trees (AST), generated from the source code of a smart contract, as used in (Momeni et al., 2019), the

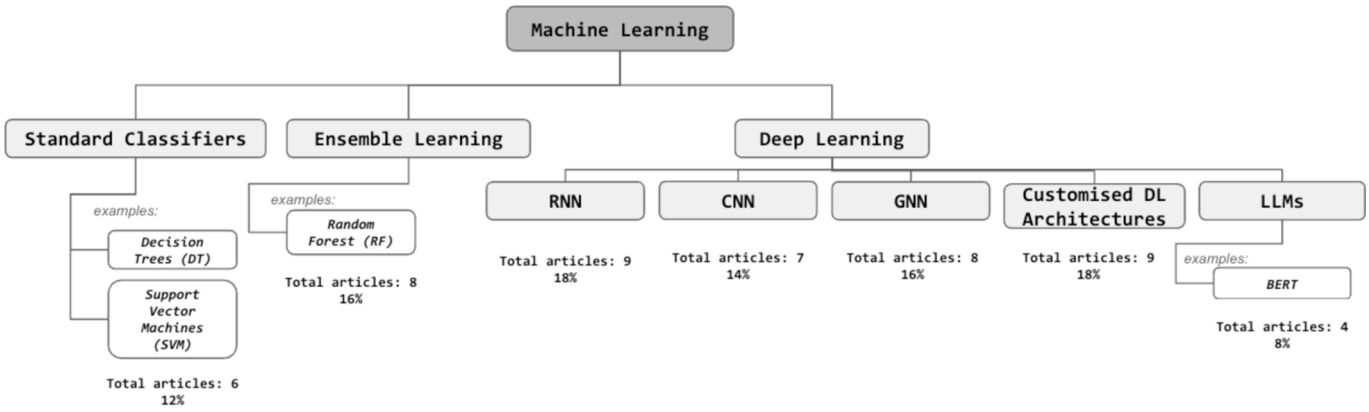


Fig. 2. Diagram illustrating the categorization used for the various Machine Learning techniques employed in this review.

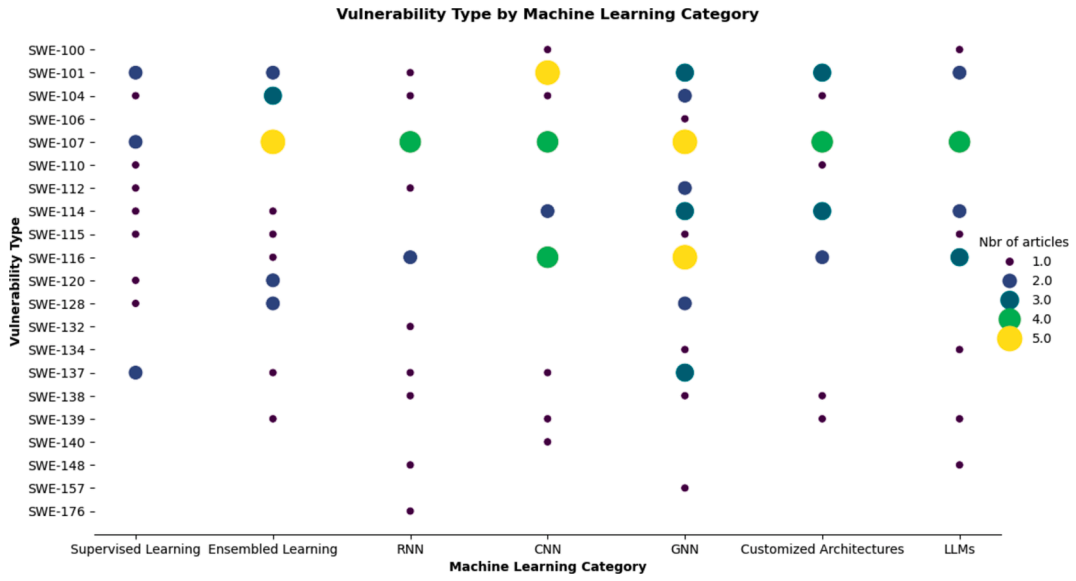


Fig. 3. Distribution of the vulnerability type following the vulnerability categorization outlined in (Zaazaa and Bakkali, 2023), by Machine Learning category.

author in (Xu et al., 2021) compares the shared child nodes between the smart contracts to be analysed and the malicious smart contracts, introducing a similarity assessment of smart contract vulnerabilities. Finally the author extracted feature vectors from those common nodes and trained a K-Nearest Neighbour (k-NN) model capable of predicting eight vulnerabilities with over 90 % accuracy, recall, and precision. Using a distinct dataset that includes only transactions involving smart contracts targeted by malicious users attempting to exploit vulnerabilities, (Mandloi and Bansal, 2022) tested Decision Trees and Random Forest classifiers to predict if a contract is vulnerable.

Focusing on feature extraction (Lohith et al., 2023) introduces a new dataset, TriPix, which represents opcodes as images. This was achieved by reading the opcodes as a stream of bytes and converting their hexadecimal equivalents into a two-dimensional matrix. The generated images underwent processing through a convolutional neural network (CNN) for feature generation. Additionally, a second dataset was created using trigram feature extraction from opcodes. Finally the authors trained a Naive Bayes method that achieved an F1-score of 99.41 %. On other hand, A two-phase Machine learning architecture used in (Mezina and Ometov, 2023). The first phase filters valid contracts using a k-NN algorithm, while the second phase focuses on identifying vulnerability types with a SVM algorithm. The resulting systems achieved a F1 score of 99.02 %, showcasing its effectiveness in enhancing smart contract security.

As observed in Table 3, features employed for training supervised classifiers were derived either from source code using the Abstract Syntax Tree (AST) representation or from opcodes using n-grams, TF-IDF (Term Frequency −Inverse Document Frequency), or word2vec (Goldberg and Levy, 2014; abs/1402.3722.). Notably, no study was found to combine both source code and opcodes for feature generation. The size of the employed datasets has shown a consistent increase over time, with a noticeable inclination towards using the SmartWild dataset. Focusing on the machine learning approaches detailed in Table 4, the highlighted (bold) method in the "Machine Learning Approach"' column represents the best-performing method when multiple algorithms were tested. Here it is clear that there are no trends within the Standard Classifiers methods. The reviewed studies illustrate a range of machine learning techniques for detecting vulnerabilities in smart contracts, utilising different feature extraction methods and classifiers. Approaches include training models on individual vulnerabilities, using Convolution Neural Networks (CNNs) with vectorized opcodes, leveraging Abstract Syntax Tree (AST) representations, and combining pre-trained models with n-gram features. The classifiers used across these studies include SVM, NN, Decision Trees, RF, k-NN, and MLP. Despite varying methods and datasets, many studies achieved high performance, with F1-scores often exceeding 90 %, demonstrating the potential of machine learning in enhancing smart contract security.

Table 3
Description of the dataset creation, including the feature identification process, dataset size, labelling approach, and, when available, the link to the data .

Paper	Dataset Composition	Size	Labelling Approach	Dataset source link
(Momeni et al., 2019)	17 Features were extracted using Abstract syntax tree (AST) representation.	1,013 Smart Contracts	Mythril Classic and Slither	Etherscan
(Liao et al., xxxx)	Features were extracted using n-grams, term frequency-inverse document frequency (TF-IDF) and word2Vec from opcodes	17,979 Smart Contracts	Oyente and Remix	Etherscan
(Xu et al., 2021)	Features were extracted using AST representation from Smart Contract source code.	3 known datasets	*	Smartbugs (GitHub, 2023a)SolifiFI-benchmark (GitHub, 2023b) SmartWild (GitHub, 2023c)
(Mandloi and Bansal, 2022)	Features extracted at transaction level.	590 sequences of transactions associated with 25 open sourced Smart Contracts.	Manual Labelling	Etherscan
(Lohith et al., 2023)	Opcodes grouped as a sequence of three and frequency of these sequence occurrences was recorded (3-gram).Opcodes used as input to generate images and extraction of pixel values.	47,398 Smart Contracts	Mythrill, Osiris, Oyente, Securify (Securify2, 2024), Slither.	SmartWild (GitHub, 2023c)
(Mezina and Ometov, 2023)	Vectorized opcodes	892, 913 Smart Contracts	MAIAN tool	BigQuery Ethereum-Blokcchain (Bigquery, 2024)

Source. (*) - not applicable, (**) - not found

Table 4
Overview of Supervised Classifier methods employed for smart contract vulnerability detection along with corresponding github link (Reproducible column) whenever applicable. This table summarises the approaches used in each article of this section.

Paper	Main Innovation	SC Vulnerabilities	Machine Learning Approach	Oversampling Technique	Reproducible	Achieved Performance
(Momeni et al., 2019)	Trained a set of machine learning models as binary classifiers for each of 46 vulnerabilities separately.	46 vulnerabilities	**SVM (support Vector Machine),**NN (Neural Networks), Decision Trees and RF (Random Forest) – as binary classifiers	None. Imbalance Training	No	F1-Score between 65 % and 99 %
(Liao et al., xxxx)	An approach to generating features using Convolution Neural Network (CNN) from vectorized opcodes with Word2Vec.	13 Vulnerabilities	**Logistic Regression**, SVM, Decision Tree, RF,and k-NN (k-Nearest Neighbors) and CNN (Convolution Neural Networks)	None. Imbalance Training.	Yes (SoliAudit, 2024)	F1-Score 90.04 %
(Xu et al., 2021)	Concept of shared child nodes between Malicious smart contracts and secure smart contracts from AST representation.	8 Vulnerabilities(SWE [101, 104, 107, 114, 120, 128, 137] and short address)	k-NN and SGD (Stochastic Gradient Descent)	*	No	Precision:95.45 %, Recal: 95.83 %
(Mandloi and Bansal, 2022)	The Dataset used in this approach has a particularity of considering only transactions with smart contracts where a malevolent user tries to exploit the smart contracts vulnerabilities.	Binary Classification	**DT** and RF	Balanced Dataset	No	F1-Score: 97 %
(Lohith et al., 2023)	Two datasets were used as inputs for standard classifiers. The first dataset employed features extracted from a CNN, using an image representation of the opcodes as input. The second dataset utilised 3-gram features extracted directly from the opcodes.	Binary Classification	**NB (Naive Bayes),** RF and k-NN	None. Imbalance Training	No	F1-Score: 99.41 %
(Mezina and Ometov, 2023)	Introduction of a two-phase detection system, leveraging customizable classifications defined to address specific smart contract security concerns.	6 vulnerabilities (SWE – [101, 107, 110, 112, 115, 137]	**k-NN & SVM,** Logistic Regression, Decision Tree and MLP (Multilayer Perceptron)	None. Imbalance Training	No	F1-Score 99.02 %

3.2. Ensemble learning Methods

One of the starting points of Ensemble Learning methods for smart contract vulnerability detection was ContractWard a multi-label classification model to tackle six vulnerabilities, encompassing Integer Overflow, Integer Underflow, Transaction-Ordering Dependence (TOD), Call Stack Depth Attack, Timestamp Dependency, and Reentrancy is presented in (Wang et al., 2021). It extracts bigram features from opcodes then eXtreme Gradient Boosting (XGBoost) model. This model boasts an average detection time of just 4 s per contract, outperforming predecessors like Oyente and Securify, which tend to be time-intensive. Inspired by ContractWard, Supriya Shakya et. al. (Shakya et al., 2022) introduces SmartMixModel, an extension that incorporates features from source code using both AST and Code2Vec. This results in Mixed-level Embedding features formed by concatenating syntactic and bytecode features. Also utilising the abstract syntax tree (AST) for feature extraction, Yang et. al. (Yang et al., 2022) training a Random Forest model to detect three specific security vulnerabilities.

Following the path of adding more features, Aljofey A. et al. (Aljofey et al., xxxx) introduces three-fold feature extraction. Using contract opcodes, transaction history, and term frequency-inverse document frequency source code features to train ensemble classifiers – Extra-

Trees and Gradient Boost algorithms adding a weighted soft voting mechanism to enhance the detection accuracy of abnormal contract accounts. In a similar fashion, the work introduced in (L et al., 2023) leverages TF-IDF and Word2Vec to train a Random Forest to detect 6 vulnerabilities.

On the other hand, fuzzing frameworks combined with ensemble learning techniques were also explored in (Eshghie et al., 2021) with the introduction of Dynamit and in (Xue et al., 2022) with the introduction of xFuzz. Dynamit consists of a Monitor, which observes transactions, and a Detector, which classifies transaction behaviour as benign or malicious. The Monitor gathers information such as gas usage, contract balance differences, and average call stack depth, while the Detector employs machine learning models, including Random Forest, to make the classification. xFuzz is a machine learning (ML)-guided smart contract fuzzing framework designed to detect cross-contract vulnerabilities. It successfully identifies 18 exploitable cross-contract vulnerabilities, demonstrating better efficiency by detecting twice as many vulnerabilities in less than 20 % of the time compared to other fuzzing tools. This framework reduces the fuzzing search space by leveraging machine learning methods to identify vulnerable paths.

Finally in (Sathiyamurthy, 2022), the author not only addressed the challenges associated with smart contracts inherent vulnerabilities but also the opacity of these contracts to non-technical stakeholders in business. The proposed solution involves an XGBoost model for smart contract vulnerability detection and the integration of a BPMN (Business Process Modeling Notation) tool to enhance overall comprehensibility.

In this section, the trend in feature generation has been utilisation of n-grams and Word2Vec/Code2Vec, both from opcodes and source code (Table 5). Notably, the Smartbugs (Liao et al., xxxx) dataset, along with Smart Contract Sanctuary (SoliAudit, 2024), emerges as one of the most frequently employed datasets. Contrary to earlier approaches, many machine learning algorithms applied here incorporate oversampling techniques (Table 6).

3.3. Recurrent Neural networks (RNN)

Utilising transaction-based information from SC accounts, a Long Short-Term Memory (LSTM) neural network is employed for anomaly detection and malicious contract identification in (Hu et al., 2021).

The articles (Goswami et al., 2021) and (Zhu et al., 2023) explore the application of LSTM neural networks to assess the vulnerability of smart contracts on the Ethereum blockchain by processing extensive sequences of opcodes. Furthermore, research (Gopali et al., 2022) beside the LSTM architecture also teste a Temporal Convolutional Network (TCN). Then the Recurrent Neural Networks evolved to bidirectional architectures. With a similar feature engineering approach, Zhang et al. (Zhang et al., 2022) introduce a Bidirectional LSTM (Bi-LSTM) neural network designed to address four types of vulnerabilities: Reentrancy, Delegatecall, Integer Overflow, and Timestamp. It achieves its best performance with an 86.40 % F1-score for the Timestamp vulnerability. The research from (Demir et al., 2023) also trained a Bi-LSTM, however, instead of using Word2Vec for opcode representation, the authors introduced op2vec. Op2vec is a vector representation specifically developed for opcodes using the skip-gram algorithm.

Following the previous trend, Jain and Tripathi et al. (Jain and Tripathi, 2023) enhance the approach by incorporating a Multi-Objective Bidirectional Gated Recurrent Unit (Bi-GRU) neural network with an attention architecture, achieving an average F1-score of 86.7 % for reentrancy and 84.4 % for timestamp vulnerabilities. This model integrates Word2Vec tokens from opcodes into a Bi-GRU, capturing contextual information bidirectionally and dynamically prioritising essential elements. In (Tereshchenko and Komleva, 2023), a very similar approach is used, employing Bi-GRU with Word2Vec tokens from opcodes, but it is trained only for reentrancy vulnerability and with a balanced dataset of 500 samples.

Unlike previous research that focused solely on opcodes or transactional data, VulHunter, introduced by Li et al. (Li et al., 2023), leverages a self-designed Bi-LSTM model to analyse both the source code and opcode of smart contracts (SCs). This methodology surpasses state-of-the-art (SOTA) approaches, achieving superior accuracy (ACC = 90.04 %, P = 87.92 %, R = 83.41 %, F1 = 85.60 %) in identifying vulnerabilities. VulHunter not only discovers vulnerabilities but also provides detailed information on defective source code statements and key opcode subsequences, aiding developers in formulating repair strategies.

Concerning the dataset composition most of the dataset were based on opcodes sequences, there is no discernible trend in the techniques employed for feature generation, and a similar observation holds for the utilisation of well-known datasets (Table 7). In the realm of recurrent neural networks, we observe an evolution from LSTMs to GRU networks, as well as a transition from unidirectional to bidirectional network architectures (Table 8).

Table 5
Description of the dataset creation for Ensemble Learning methods, including the feature identification process, dataset size, labelling approach, and, when available, the link to the data .

Paper	Dataset Composition	Size	Labelling Approach	Dataset
(Wang et al., 2021)	Extract Features using n-grams from opcodes	49,502 Smart Contracts	Oyente	Ethereum website
(Eshghie et al., 2021)	Account Features (transactions and balances)	25 Smart Contracts (monitored 105 transactions)	*	Etherscan
(Xue et al., 2022)	– 20 extract features from bytecode using Word2Vec, – 7 extracted features from CFG.	100,000 Smart Contracts	Solhint, Slither and Securify	Etherscan& Smart Contract Sanctuary (GitHub, 2023d)
(Aljofey et al., xxxx)	Extracts features from opcodes (n-grams), transaction history, and the source code (TF-IDF) of the contract account.	1,904 Smart Contracts	Labelled from Etherscan.io	Own dataset in (GitHub, 2023e)
(Shakya et al., 2022)	– High-level syntactic features: abstract syntax tree (AST) and Code2Vec, – Low-level Features: n-grams from opcodes (Wang et al., 2021).	70, 000 Smart Contracts	SmartCheck (GitHub, 2023f)	Smart Contract Sanctuary (GitHub, 2023d)
(Sathiyamurthy, 2022)	N-grams features extracted from source code	1,388 Smart Contracts	**	Smartbugs (GitHub, 2023a)
(Yang et al., 2022)	Sub-ASTs, outlining functions, state variables, and function modifiers, are transformed into sequences and vectors through the utilisation of preorder traversal and word embedding techniques.	3,000 Smart Contracts	Slither	Etherscan
(Singh and Chakravarthi, 2023)	– TF-IDF feature extraction from source code (solidity files). – Word2Vec or Skip-gram combined with CBOW feature extraction from source code.	**	*	SmartBugs (GitHub, 2023a)

Source. (*) - not applicable, (**) - not found

Table 6
Overview of Ensemble Learning methods employed for smart contract vulnerability detection along with corresponding github link (Reproducible) whenever applicable. This table summarises the main approach used in each article.

Paper	Main Innovation	SC Vulnerabilities	Machine Learning Approach	Oversampling Technique	Reproducible	Achieved Performance
(Wang et al., 2021)	Extracted bigram features from opcodes, applied undersampling techniques and used eXtreme Gradient Boosting (XGBoost) as the main classifier.	6 Vulnerabilities (SWE [101, 107, 114, 116, 139])	**XGBoost**, AdaBoost,RF, SVM and k-NN as comparison models.	SMOTE and SMOTETomek	Partial. (ContractWard, xxxx)	F1-Score: 96,41 %
(Eshghie et al., 2021)	A transactional dataset where the author manually generates the interactions with the selected smart contracts and uses these transactions as his raw data.	1 Vulnerability (SWE-107)	**RF**, NB, Logistic Regression, k-NN, SVM, NN.	Balanced dataset	No	F1-Score: 86 %
(Xue et al., 2022)	A framework for detecting cross-contract vulnerabilities is developed, leveraging a fuzzing approach integrated with machine learning methods to expedite the identification of vulnerability paths.	18 Vulnerabilities	**Easy Ensemble Classifier,** XGBoost, Decision Trees, SVM, k-NN, NB, Logistic Regression	None. Imbalance Training	No	F1-Score: 40.83 %
(Aljofey et al., xxxx)	From 3-fold features a Extra-trees classifier(ETC) and gradient boosting machine (GBM) a binary classifier is trained.	Binary Classification	ETC (Extra-trees classifier), GBM (Gradient Boosting Machine)	ADASYN	Yes (GitHub, 2023e)	F1-Score: 88.74 %
(Shakya et al., 2022)	2-fold features to train an XGBoost classifier.	10 vulnerabilities	XGBoost	SMOTE for over-sampling	No	F1-Score: 98.20 %
(Sathiyamurthy, 2022)	A bi-gram feature extraction from smart contract code snippets is used to train a XGBoost classifier for 4 classes of vulnerabilities: Denial-of-Service, unconditional external call, reentrancy and transaction origin.	4 Vulnerabilities (SWE [104, 107, 115, 128])	**XGBoost,** RF, k-NN, SVM	None. Balanced Training	No	F1-Score: 97.10 %
(Yang et al., 2022)	An approach that segments the AST into sub-ASTs based on functions, state variables, and function modifiers and then uses Word2Vec to extract feature vectors.	3 Vulnerabilities (SWE [104, 107, 120])	**RF**	None. Imbalance Training	No	Avg F1-Score: 88.50 %
(J J L, Singh K, Chakravarthi B. , 2023)	Extracting features from source code using not only Word2Vec but also Skip-gram and CBOW nlp approaches.	6 vulnerabilities(SWE [101, 104, 107, 120, 128, 137])	**RF,** Decision Trees, Bagging, AdaBoost & GradientBoost	SMOTE for over-sampling	No	F1-score:83.12 %

Table 7
Description of the dataset creation for RNN algorithms, including the feature identification process, dataset size, labelling approach, and, when available, the link to the data .

Paper	Dataset Composition	Size	Labelling Approach	Dataset source
(Hu et al., 2021)	14 Statistical features from transactions	– Transaction data from Jul 2015 to Feb 2018 – 4,593 Smart Contracts	*	Etherscan
(Goswami et al., 2021)	Converted Bytcodes to opcode using a disassembler tool.	165,652	Maian and Mythril	(Google Cloud, 2023)
(Gopali et al., 2022)	Opcodes encoded with a tokenizer.	892, 913 Smart Contracts	MAIAN tool	(Bigquery, 2024)
(Zhu et al., 2023)	Vectorized opcode sequences by assigning a number between 0 and 142, considering that the total number of existing opcodes in Ethereum is 142.	**	Geth (short for "Go-Ethereum") client interface.	Etherscan
(Zhang et al., 2022)	Vectorized opcode sequences.	**	**	**
(Tereshchenko and Komleva, 2023)	Word2Vec vector representation from opcodes.	250 smart contracts with Reentrancy vulnerability, 250 non-vulnerable smart contracts	**	SmartWild (GitHub, 2023c)
(Jain and Tripathi, 2023)	Word2Vec vector representation from opcodes.	11,000	*	SoliAudit (SoliAudit, 2024)
(Demir et al., 2023)	Analogous to Word2Vec a Op2Vec vector representation was created from 15,000 SCs using Skip-grams algorithm.	From 15,000 initial smart ontracts only 248 smart contracts were used to train the model.	Slither	**
(Li et al., 2023)	From bytecode and opcode a CFG is extracted	>200,000 Smart contracts	SmartFast, Slither and Oyente	Own dataset (VulHunter, 2024)

Source. (*) - not applicable, (**) - not found

3.4. Convolutional Neural networks (CNN)

Targeting Short Address, Integer Overflow and Underflow, and Locked money issues, a Convolutional Neural Network – Convolution Neural Network Based on Slice Matrix (CNNBOSM) is trained using opcode sequences. These sequences are divided by the opcode "RETURN'' (bytecode – 0xF3), serving as the segmentation point to create a slicing matrix (Xing et al., 2020). The author also developed two other vulnerability detection models, including Neural Network Based on Opcode Feature (NNBOOF) and Random Forest Based on Opcode Feature (RFBOOF). Experimental results indicate that the slicing matrix significantly enhances the accuracy of vulnerable contract identification, outperforming neural network models relying solely on opcode features. However, the random forest-based model emerges as the most effective in detecting smart contract vulnerabilities. Then, Yuhang Sun et. al. (Sun et al., 2021) added a self-attention mechanism to a

Table 8
Overview of RNN methods employed for smart contract vulnerability detection along with corresponding github link (Reproducible column) whenever applicable. This table summarises the main approach used in each article of this section. (*) – not applicable, (**) – not found.

Paper	Main Innovation	SC Vulnerabilities	Machine Learning Approach	Sampling Technique	Reproducible	Achieved Performance
(Hu et al., 2021)	LSTM with transactional data features for anomaly-detection purposes.	Unsupervised Learning	LSTM	*	No	F1-Score: [82.7 % – 96.5 %]
(Goswami et al., 2021)	LSTM with features from opcodes.	3 Vulnerabilities (SWE-[137, 138])	LSTM	Balanced dataset	No	F1-Score: 93.00 %
(Gopali et al., 2022)	A Temporal Convolutional Network (TCN) model was tested using tokenized opcodes.	Binary Classification	**Temporal Convolutional Network (TCN)**, LSTM	SMOTE	No	F1-Score: 94.83 %
(Zhu et al., 2023)	LSTM trained with opcodes sequences to detect 7 types of vulnerabilities	7 Vulnerabilities (SW [104, 107, 115, 116, 132, 148, 176])	LSTM	**	No	F1-Score: 79.74 %
(Zhang et al., 2022)	A Bidirectional LSTM network is trained to identify 4 types of vulnerabilities from opcode sequences.	4 Vulnerabilities (SWE – [101, 107, 112, 116])	Bi-LSTM	**	No	Best F1-Score: 86.40 %
(Tereshchenko and Komleva, 2023)	Bi-GRU with attention mechanism machine learning model using Word2Vec vector representation from opcodes to detect exclusively Reentrancy vulnerability type.	1 Vulnerability (SWE-107)	Bi-GRU with attention mechanism.	Manually Undersampled	No	F1-Score: 91.41 %
(Jain and Tripathi, 2023)	Bi-GRU with attention mechanism machine learning model using Word2Vec vector representation from opcodes.	2 vulnerabilities(SWE-[107, 116])	Bi-GRU with attention mechanism.	None. Imbalance Training	No	Reentrancy F1-Score: 86.70 % Time Stamp F1-Score: 84.40 %
(Demir et al., 2023)	First a Op2Vec vector representation was created from 15,000 SCs using Skip-grams algorithm.Then, a Bidirectional LSTM network was trained to detect if a Smart Contract is vulnerable or not.	Binary Classification	Bi-LSTM	Manually balanced Dataset	No	Accuracy: 96 %:
(Li et al., 2023)	VulHunter: Bi-directional LSTM (Bi2-LSTM) model, trained with source code and opcodes features.	30 Vulnerabilities	Bi-LSTM	None. Imbalance Training	Yes (VulHunter, 2024)	F1-Score: 85.60 %

Convolutional Neural Networks (CNN), to efficiently identify similar states within the network, focusing in detecting 3 types of vulnerabilities: Reentrancy, Arithmetic Issues(Integer Overflow/Underflow) and Time manipulation (Timestamp Dependence). Instead of using opcodes to train a CNN, Zhou Q. et al. (Zhou et al., 2022), choose the representations of smart contracts derived from abstract syntax trees (AST) to create a word embedding matrix using Word2Vec that fed a CNN followed by a pooling and hidden layer to predict the vulnerability types: Non-determinism, Reentrancy, Exception State, Integer Overflow and Underflow.

Adding a backpropagation process upon the work of (Sun et al., 2021), Zhang L. et al. (Zhang et al., 2022) trained a CNN called the Multiple-Objective Detection Neural Network (MODNN). A smart contract vulnerability detection tool that can inspect 12 types of vulnerabilities and also identify unknown threats without the need for specialist or predefined knowledge. This paper also developed a data processing tool called Smart Contract-Crawler (SCC). Experiments showed that MODNN achieved an average F1 Score of 94.8 %.

Another feature engineering approach, utilising bi-grams and TF-IDF feature extraction, was employed by Zhendong Wu et al. (Wu et al., 2022). They developed a set of deep learning models to classify six smart contract vulnerabilities, achieving the best performance with the ResNets model. In a similar vein, He et. al. (He et al., xxxx) applied the same feature engineering to create a framework called FSL-Detect. However this framework addresses the challenge of limited unknown threat samples, employing CNN with *meta*-learning MAML (Finn, 2017). Following the *meta*-learning approach, (Yang et al., 2023) introduced the SCLMF (Small-sample Classification Learner-Meta-Learner Framework) which also adopts a learner-*meta*-learner architecture, utilising bytecode, RGB image generation, and a convolutional neural networks. The author further explored another framework, the SCSVM (Smart Contract Support Vector Machine), leveraging support vector machines in conjunction with Word2Vec.

Regarding the features employed for CNN architectures, similar techniques and well-known datasets mentioned in the previous sections are applied here (Table 9). However, an incremental complexity is observed in the architecture of the machine learning frameworks used in this section (Table 10), where CNNs are combined with attention network layers or with *meta*-learning mechanisms such as MAML (Finn, 2017).

3.5. Graph Neural networks (GNN)

In (Zkik et al., 2023) the authors leverage feature extraction from code syntax analysis to explore three different Graph Neural Networks architectures: Graph convolutional neural networks (GCNs), graph isomorphism networks (GIN), and graph attention layers (GAT), to detect three types of smart contract vulnerabilities. Then a Heterogeneous Graph Transformer Networks (S_HGTNs) proposed by Liu L. et. al. (Liu et al., 2022), combining the output matrix from a transformer neural network with GCNs, was trained as a binary classifier.

EtherGIS, as presented by Qingren Zeng et al. (Zeng et al., 2022), utilises a fusion of graph neural networks (GNN) and expert knowledge. This framework leverages the extraction of crucial graph features from control flow graphs (CFGs) generated from opcodes. What sets EtherGIS apart is its capacity to construct refined EVM instruction corpora, facilitating the pinpointing of vulnerable code segments. Also in (Liu et al., 2023) we have another approach combining graph neural networks (GNN) with expert rules, but this time adding a detection and blocking mechanism at the EVM level.

MANDO (Nguyen et al., 2022) employs a novel architecture, the Heterogeneous Graph Neural Network with Node-Level Attention, utilising both control-flow graphs and call graphs to create embeddings for various levels of granularity in SCs. This model effectively detects vulnerabilities at both line and contract levels. The approach involves a four-component system, initially processing Solidity source code to

Table 9
Description of the dataset creation CNN algorithms, including the feature identification process, dataset size, labelling approach, and, when available, the link to the data .

Paper	Dataset Composition	Size	Labelling Approach	Dataset source link
(Xing et al., 2020)	Split the opcode into segments, creating what the author called slice matrix.	19,145 Smart Contracts	*	*
(Sun et al., 2021)	Word2Vec vector representation from opcodes.	8,632 Smart Contracts	Oyente and Mythril	Etherscan
(Zhou et al., 2022)	From the abstract syntax tree (AST) and using Word2Vec, an embedding matrix was generated as the feature space.	10,567	Mythril	Etherscan
(Wu et al., 2022)	Features extracted using Mean term frequency-inverse document frequency (TF-IDF) and n-grams from opcodes (Wang et al., 2021).	**	Mythril and Oyente	**
(Zhang et al., 2022)	− A pre-trained Bert model is used to convert the crucial operation sequence (COS) from opcodes. COS is a concept introduced here. − Co-occurrence matrices based on the co-occurrence countsof operation codes.	18,796 Smart Contracts	Mythril, Smartcheck and Osiris	Own Dataset (MODNN, 2024)
(He et al., xxxx)	From bytecode and opcode a CFG is extracted using n-gram and TF-IDF to extract features from the graph representation.	19,998 Smart Contracts	Oyente, Slither	Own dataset, no source info.
(Yang et al., 2023)	2 Frameworks: − Word2Vec from source code − **SCSVM**. − learner-*meta*-learner architecture utilising smart contract bytecode − **SCLMF**.	2 known Datasets	Mythrill, Osiris, Oyente, Securify (Securify2, 2024) and Slither.	SmartBugs (GitHub, 2023a)ScrawID (VulHunter, 2024)

Source. (*) - not applicable, (**) - not found

Table 10
Overview of CNN algorithms employed for smart contract vulnerability detection along with corresponding github link (Reproducible column) whenever applicable. This table summarises the main approach used in each article of this section.

Paper	Main Innovation	SC Vulnerabilities	Machine Learning Approach	Sampling Technique	Reproducible	Achieved Performance
(Xing et al., 2020)	Using opcode sequences splitted by opcode " RETURN" to create a Slice matrix. Then using CNN and RF methods to detect three types of vulnerabilities	3 Vulnerabilities (SWE-[101, 137] and Short Address)	**CNN and RF**	None. Imbalance Training.	No	F1-Score between [85 % − 95 %] according to the vulnerability type.
(Sun et al., 2021)	From the Word2Vec embeddings (from opcodes), the author trained a CNN with a Self Attention layer to extract characteristics of words connected in long distance.	3 Vulnerabilities (SWE-[101, 107, 116])	**CNN** trained with a **Self Attention** Layer − as a binary classifier.	None. Imbalance Training	No	F1-Score between [86 % − 90 %] according to the vulnerability type.
(Zhou et al., 2022)	Tree-based machine learning vulnerability detection (TMLVD) framework: a tree-based Convolution Neural Network (CNN) trained with Word2Vec embedding matrix.	4 Vulnerabilities (SWE-[100, 101, 107, 140])	Tree-based CNN	None. Imbalance Training	No	F1-Score: between 78.60 % and 98.12 % according to the vulnerability type.
(Wu et al., 2022)	Employing bi-grams and TF-IDF feature extraction from opcodes for training deep learning models and evaluating performance across six vulnerability types.	6 Vulnerabilities (SWE-[101, 107, 114, 116, 139])	**ResNets (**CNN), LSTM, CNN, BiLSTM.	None. Imbalance Training	No	F1-Score: 82 %
(Zhang et al., 2022)	A CNN with the attention mechanism and backpropagation process was developed to detect 12 types of vulnerabilities, called Multiple-Objective Detection Neural Network (MODNN)	12 types of Vulnerabilities	**MODNN** − CNN with the attention mechanism, tested against DR-GCN, Eth2Vec (Ashizawa et al., 2021) and SoliAudit (Liao et al., xxxx).	Focal loss algorithm to balance the dataset	No	Best F1-Score: 96.28 % (own dataset)
(He et al., xxxx)	CNN with a *meta*-learning approach focusing on detected unknown vulnerabilities.	2 Vulnerabilities (SWE-[107,116])	CNN with Meta-Learner MAML (Finn, 2017)	None. Imbalance Training	No	F1-Score between 90.94 % and 93.98 %
(Yang et al., 2023)	Application of an algorithm for *meta*-learning, where it can solve new learning tasks using only a small number of training samples − Learner-Meta-learner Framework (SCLMF).	4 Vulnerabilities (SWE-[101, 104, 114, 116])	SVM, **CNN** with Meta-Learner MAML (Finn, 2017)	None. Imbalance Training	No	SCSVM: F1-score between 87.51 % and 87.68 % SCLMF:

generate heterogeneous graphs at both contract and line levels, then a Multi-Metapaths Extractor handles dynamic types in meta paths, while the Multi-Level Graph Neural Networks use a Topological Graph Neural Network for graph topology and a Node-Level Attention Heterogeneous Graph Neural Network to weigh metapath importance, generating embeddings for coarse-grained and fine-grained vulnerability detection. The Two-Phase Vulnerability Detector classifies contracts and, for vulnerable ones, performs node classification to identify specific vulnerable locations in the code, MANDO-GURU (Nguyen et al., 2022) is a end user tool employing the MANDO approach. MANDO-HGT (Nguyen et al, xxxx) is an evolution of MANDO employing almost the same architecture but changing the Meta-path extractor to the Mate-Relation Extractor.

Another noteworthy method highlighted in (Hobor and Abdelaziz, 2023) is the Deep Learning Vulnerability Analyzer (DLVA). Constructed with a three-component architecture, DLVA includes the Smart Contract to Vector (SC2V) engine, utilising Node2Vec for feature extraction from Control Flow Graphs (CFGs) representation, thereby mapping bytecode to a high-dimensional vector space. This is followed by a Graph Convolutional Neural Network (GCNN), then add two classifiers, Sibling

Detector (SD) based on vector proximity and Core Classifier (CC) based on neural networks. DLVA achieved an accuracy of 92.7 %. In addition to DLVA's feature engineering, GVD-net, (Wang et al., 2022) expands its feature space by generating a non-complete random walk sequence from CFG It then also employs a similarity analysis to classify vulnerabilities like arithmetic issues, access control, or asset freezing.

The majority of the work produced here used feature generation based on the Control Flow Graph (CFG) representation of the source code (Table 11). Additionally, the Smartbugs dataset is one of the most explored for GNN. From Table 12, it is evident that advanced techniques such as Heterogeneous Graph Transformer Networks (HGTNs), Graph Neural Networks (GNNs) including GCNs, GIN, and GAT, alongside innovative frameworks like MANDO and DLVA, exhibit robust capabilities in identifying vulnerabilities across diverse types and complexities. These methods harness sophisticated graph-based and deep learning models, achieving F1-scores ranging from 76.32 % to 95.99 % in classification tasks. Despite challenges such as imbalanced training data, there is a clear trend of increasing complexity and performance improvement observed from earlier studies to the latest advancements (Table 12). Furthermore, there is a trend towards designing machine learning frameworks to address fewer vulnerabilities concurrently, leading to a specialisation of these models.

3.6. Customised deep learning architecture Methods

In (Xu et al., 2023) a TextCNN model to detect 12 types of smart contract vulnerabilities is presented, trained on opcode sequences, which were vectorized using the Skip-Gram algorithm. To enhance the model's robustness against redundant or non-functional code segments, random non-essential opcodes were deliberately inserted into the smart contracts. Jie et. al. (Jie et al., 2021) also leverages a TextCNN architecture. The author first introduces a comprehensive full-stack feature engineering. This comprises an integration of the source code layer, and Ethereum virtual machine (EVM) bytecode layer creating a multimodal feature space. Then combining convolutional neural networks (TextCNN) with graph convolutional neural networks (GCNs), to extract features from those layers. Finally a binary classifier to detect Smart contract vulnerabilities is training using a bidirectional long short-term memory (Bi-LSTM) with a self-attention mechanism and a conditional random field (CRF) layer. Also a custom deep learning architecture with fusion of multimodal features is presented in (Deng et al., 2023). Another ensemble learning approach, SCVDIE-ENSEMBLE is introduced by Zhang et. al. (Zhang et al., 2022), exploring seven different neural networks (NNs) pretrained on information graphs to detect 7 vulnerability types.

A smart contract debugger focused on reentrancy vulnerability is introduced in (Zhang et al., 2023). ReVulDL is a unified pipeline, utilising a graph-based pre-training model to analyse propagation chains for vulnerability detection and interpretable machine learning for statement-level localization.

Two similar approaches using Transformer architectures to detect vulnerabilities in smart contracts as a binary classification problem are introduced in (Balcı et al., 2023) and (Gong et al., 2023). The first approach used vectorized opcodes sequences as input features, achieving a F1-score of 86 %. Subsequently, in (Gong et al., 2023), the author utilised a CFG representation from opcodes to achieve a F1-Score of 95.17 %. Additionally in (Zuo et al., 2023), using the same type of input features as (Gong et al., 2023), a self-supervised learning (SSL) model, consisting of an encoder and discriminator, is trained on a discriminative task of detecting vulnerable smart contracts. The encoder is a combination of a Transformer and a CNN. While the discriminator is a similarity calculator between the input and the output of the encoder. This architecture reaches a F1-Score of 87.60 %. In (Yuan et al., 2021), a method is presented that leverages source code snippets, contrasting with previous transformer model approaches. As mentioned, this method starts by extracting features using Word2Vec from specific code snippets and then training a transformer neural network. This approach achieved an F1-score of 94.82 % in classifying smart contracts as either vulnerable or secure.

Here, we observe the combined application of all feature generation methods used in the previous articles. Additionally, there is no distinct trend in the choice of well-known datasets in this field (Table 13). As anticipated, the machine learning approaches here are more complex, with several frameworks combining CNN with GNN and RNN, leading to some of the best performances achieved in the realm of smart contract vulnerability detection (Table 14).

Table 11
Description of the dataset creation for GNN, including the feature identification process, dataset size, labelling approach, and, when available, the link to the data .

Paper	Dataset Composition	Size	Labelling Approach	Dataset source link
(Liu et al., 2022)	– Account Features (Transactions, Gas, Fees and Blocks info) – Code features using Doc2Vect	1,251 secure SCs and 131 vulnerable SCs	*	Dataset from paper (Bartoletti et al., 2020)
(Zkik et al., 2023)	– Feature extracted from syntax analysis and keyword matching.	4,170 Smart Contracts	**	Vntchain smart contacts: (Chain, 2020)
(Zeng et al., 2022)	Control Flow Graph from source code	9,000 secure SCs and 1,000 vulnerable SCs.	Oyente, Mythril	Own dataset, no source info.
(Nguyen et al., 2022)	With the Slither tool processing the source code of each input Ethereum smart contract, two types of features were then generated: heterogeneous control-flow graphs (HCFGs) and heterogeneous call graphs (HCGs).	2,742 Clean SCs and 493 vulnerable SCs	Manticore, Mythril, Oyente, SecurifySlither (Kushwaha et al., 2022) and Smartcheck	Smartbugs (GitHub, 2023a)SolifiFI-benchmark (GitHub, 2023b)SmartWild (GitHub, 2023c)
(Hobor and Abdelaziz, 2023)	The feature extraction process involves converting bytecode to opcodes, analysing the control flow of the program through a CFG, and then extracting features from the CFG nodes using N2V.	113,166 Smart Contracts	Slither	Ethereum SC large (Ethereum, 2024) Ethereum SC small (EthereumSC, 2024)
(Nguyen et al, xxxx)	Build CFG and Call Graphs (CGs) from source code and opcodes and convert them into heterogeneous forms, called heterogeneous control-flow graphs and heterogeneous call graphs, and fuse them into heterogeneous contract graphs (HCGs).	55,000 Smart Contracts	Slither, EtherSolve	Dataset (MANDO, 2024a) derived from SmartBugs (GitHub, 2023a) and SolidFI-benchmark
(Liu et al., 2023)	– contract code's data flow and control flow relationships from Source code. – graph normalisation using the temporal message propagation network model (TMP)	40,932 Smart Contracts	SmartCheck, Oyente, MythX,Securify2.0, and Slither	Etherscan
(Wang et al., 2022)	– Control Flow Graph from source code – Node2Vec from source code	2 known datasets	Oyente and Manticore	SmartBugs (GitHub, 2023a) and SmartWild (GitHub, 2023c)

Source. (*) - not applicable, (**) - not found

Table 12
Overview of GNN algorithms employed for smart contract vulnerability detection along with corresponding github link (Reproducible column) whenever applicable. This table summarises the main approach used in each article of this section. (*) – not applicable.

Paper	Main Innovation	SC Vulnerabilities	Machine Learning Approach	Sampling Technique	Reproducible	Achieved Performance
(Liu et al., 2022)	From account features and smart contract code features an Heterogeneous Graph Transformer Network is applied to detect if a contract is vulnerable or fraudulent.	Binary Classification	**Heterogeneous Graph Transformer Networks (S_HGTNs)**, Comparison models: SVM, CNN,RNN, GCN, GAT.	None. Imbalance Training	No	F1-Score: 82 %
(Zkik et al., 2023)	Evaluation of 3 GNN approaches to adresse 3 types of vulnerabilities using code syntax features.	3 Vulnerabilities (SWE-[107, 116, 157])	Graph convolutional neural networks (GCNs), **graph isomorphism networks (GIN)**, and graph attention layers (GAT)	None. Imbalance Training	No	GIN F1-Score: 76.32 %
(Zeng et al., 2022)	EtherGIS is a vulnerability detection framework that implements graph neural networks (GNN) trained with features from CFG representation.	6 Vulnerabilities (SWE-[106, 107, 112, 114, 115, 116])	GNN	SMOTE and SMOTETomek	No	F1-Score: 80.50 %
(Nguyen et al., 2022)	MANDO: a smart contract vulnerability detection, providing comprehensive results for seven distinct vulnerabilities including a line-level vulnerability identification.	7 Vulnerabilities(SWE-[101, 104, 107, 114, 116, 128, 137])	**HGNN (Heterogeneous Graph Neural Network),** baseline GCN	None. Imbalance Training	Yes (MANDO, 2024b)	Time Manipulation F1-Score: 90.29 %
(Hobor and Abdelaziz 2023)	Deep Learning Vulnerability Analyzer (DLVA) leverages N2V, GNN and FFN.	29 Vulnerabilities (Binary Classifier)	GNN combined with Feed Forward Network (FFN) and a similarity process.	None. Imbalance Training	Partial, Docker image: (DLVA, 2024)	Accuracy 92.70 %
(Nguyen et al, xxxx)	The MANDO-HGT is composed of: Heterogeneous Contract Graph Generator, Meta Relations Extractor, Node Features Extractor, MANDO-HGT Graph Neural Network, and Two-Phase Vulnerability Detector.	7 Vulnerabilities (Nguyen et al., 2022)	HGT (Heterogeneous Graph Transformer)	None. Imbalance Training	Yes (MANDO, 2024a)	Time Manipulation F1-Score: 95.99 %, Reentrancy F1-Score: 94.78 %
(Liu et al., 2023)	A smart contract vulnerability detection framework based on GNN models to detect vulnerabilities during deployment, combined with expert rules, working at the EVM level.	4 vulnerabilities (SWE-[107, 112, 116, 134])	GNN combined with Expert Rules	None. Imbalance Training	No	F1-Score between 84.10 % and 87.34 %
(Wang et al., 2022)	GVD-net: from a generated non compete random walk sequence, and using graph embedding algorithm – Node2Vec – a 256-dimensional vector and gain weight matrix are generate, then by similarity analysis a given source code is tested and considered vulnerable if the similarity is higher that a given threshold.	3 Vulnerabilities (SWE-[101, 137, 138])	A similarity analysis between non-Euclidean graph and random walk sequences	*	No	Accuracy between 89.70 % and 91.30 %

3.7. Large language models (LLMs)

Eth2Vec is a tool focused on security analysis of the Solidity language as a target of static analysis of Ethereum smart contracts aiming to detect the existence of vulnerabilities and classify the kinds of vulnerabilities in the codes to be analysed, proposed by Ashizawa N. et al. (Ashizawa et al., 2021). The Eth2Vec is a neural network for natural language processing that automatically learns features of vulnerable Ethereum Virtual Machine (EVM) bytecodes. Eth2Vec can detect vulnerabilities in smart contracts by comparing the code similarity between target EVM bytecodes and the EVM bytecodes it already learned. Eth2Vec leverages the PV-DM model (Jin et al., 2015) which vectorizes each paragraph by taking into the context of the paragraph account. However, this framework is unsupervised so the performance of Eth2Vec can be improved by utilising supervised learning. This is explored in (Duan et al., 2023) where a BERT language model is employed to extract features from source code and then combine them with bi-gram features from opcodes to train 5 different classifiers. A different usage of BERT model is introduced with ASSBert in (Sun et al., 2023), where the author addressed the challenge of limited labelled data using a bidirectional encoder representation from transformers network that combines active and semi-supervised learning methodologies.

The paper (Storhaug et al., 2023) is on the periphery of our research scope, but its content is worth mentioning. An SC out-complete tool is introduced here by fine-tuning a GPT-J model that is then used as a Smart Contracts synthesiser, mitigating the generation of vulnerable code.

In this section, it is noteworthy that the most recent studies in this field have shifted focus slightly to highlight vulnerable sections of the code and tools for auto-completion, aiding the community in writing safer code during development. This approach has a higher impact, and efforts should be continued in this direction. The datasets and feature engineering techniques applied here align with those previously discussed (Table 15). However, some of the machine learning approaches used here require different evaluation metrics, such as BLEU (Mathur et al., 2020), to understand the quality of the auto-generated code (Table 16).

4. Discussion

This paper systematically reviews the application of machine learning (ML) methods to detect vulnerabilities in smart contracts on the Ethereum blockchain. The key findings indicate that diverse ML techniques, spanning from supervised learning to advanced deep learning architectures, exhibit significant potential in identifying and mitigating smart contract vulnerabilities. The review is structured into seven main categories of ML methods: supervised learning, ensemble learning, recurrent neural networks (RNN), convolutional neural networks (CNN), graph neural networks (GNN), customised deep learning architectures and LArge Language Models (LLMs).

Table 13
Description of the dataset creation for the customised deep learning architectures, including the feature identification process, dataset size, labelling approach, and, when available, the link to the data .

Paper	Dataset Composition	Size	Labelling Approach	Dataset source link
(Xu et al., 2023)	Vector representation of Opcodes using the skip-gram algorithm (Mikolov et al., 2013)	60,000 unlabelled SCs and 17,000 labelled SCs	**	(Website. Available: starStraw., 2022)
(Jie et al., 2021)	– Extracted features from bytecodes with control flow graphs (CFG) (EVMB), – Extracted features using CFG with the output of a static analysis tool from source code (Build base – BB), – Extracted features from source code using Word2Vec and BERT (Source Code – SC).	5,000 Smart Contracts	Slither	Training data from Eth2Vec (GitHub, 2023g)
(Zhang et al., 2022)	Extracts features from opcodes (removes all opcodes of Stack type), 46 different opcodes were used. Then critical opcodes sequences, identified by predefined rules, are used as input data to construct an infographic (IG) based on co-occurrence frequency matrix.	21,667 Smart Contracts	Oyente, Mythril, and Securify	DEDAUB
(Zhang et al., 2023)	Convert source code into an abstract syntax tree (AST), extract data flow relationships, and construct propagation chains from the extracted relationships.	47,398	Mythrill, Osiris, Oyente, Securify (SoliAudit, 2024; GitHub, 2023g), Slither.	SmartWild (GitHub, 2023c)
(Balcı et al., 2023)	Vectorized Opcodes.	From 16,363 SCs 1202 non vulnerable and 713 greedy SCs were selected.	MAIAN tool	Etherscan
(Gong et al., 2023)	Features extracted from Control Flow Graph (CFG) using Opcodes.	2,273 SCs,	(**) – not found	Etherscan
(Zuo et al., 2023)	Features extracted from Control Flow Graph (CFG) using Opcodes.	1,000	*	Etherscan
(Yuan et al., 2021)	Features extracted from Source code using Word2Vec.	40,000 Smart Contracts	Oyente	Smartbugs (GitHub, 2023a)
(Deng et al., 2023)	– N-grams and Word2Vec from opcodes; – Fast text and Word2Vec from Source code; – GCN feature generation using CFG from source code.	9,252	*	ScrawID (Scraw, 2024)

Source. (*) - not applicable, (**) - not found

Here we will thoroughly investigate the three primary research questions outlined in this paper. This exploration will delve into existing methods and techniques related to each question, offering valuable insights for both researchers and practitioners.

RQ1. What is the taxonomy of machine learning methods used for smart contract vulnerability detection in Ethereum Blockchain and what are the strengths and limitations of these methods in identifying vulnerabilities?

Several machine learning approaches can be employed in the realm of smart contract vulnerability detection, encompassing supervised learning, unsupervised learning and active learning. Presently, the majority of machine learning-based methods for smart contract vulnerability detection predominantly leverage supervised learning. There's an evident trend towards increased complexity, particularly with the use of deep learning architectures. However, this complexity doesn't consistently translate to substantial performance gains, and in some cases, it has negligible impact.

Supervised learning methods, including SVM, NN, Decision Trees, and Random Forest demonstrate high accuracy and efficiency in detecting vulnerabilities. However, in some cases, such as (Liao et al., xxxx) and (Lohithet al., 2023), a CNN is used for feature extraction, blendy supervised classifiers with deep learning architectures. Ensemble learning methods, such as XGBoost and Random Forest combined with sophisticated feature extraction methods (e.g., n-grams, AST, TF-IDF), show improved accuracy. Notable models like ContractWard (Wang et al., 2021) and SmartMixModel (Shakya et al., 2022) effectively handle multiple vulnerabilities. Recurrent neural networks (RNN), particularly LSTM and its variants (e.g., Bi-LSTM, Bi-GRU), are applied to sequence data from smart contracts, achieving robust performance, especially for vulnerabilities like reentrancy and timestamp dependencies. Furthermore, ensemble learning methods often combine oversampling techniques to achieve high performance, which may lead to overfitting the data and losing the capability of generalisation.

Convolutional neural networks (CNN), often enhanced with mechanisms like self-attention or combined with *meta*-learning, have shown high effectiveness in vulnerability detection, with some models like MODNN (Zhang et al., 2022) achieving F1-scores up to 94.8 %. Graph neural networks (GNN), including GCNs, GIN, and GAT, leverage graph-based features generated from control flow graphs, abstract syntax trees and vectorized opcodes, have shown promise in detecting complex vulnerabilities with high precision.

Advanced customised deep learning architectures, which integrate components from CNNs, GNNs, RNNs, and transformers, represent the cutting edge in smart contract vulnerability detection. These architectures achieve some of the highest performance metrics, particularly in detecting less conventional smart contract vulnerability types, such as SWE-106 and SWE-134 (see Fig. 3).

Also from Fig. 3, GNN is the category that has been used to address the most variety of smart contract vulnerabilities. It also achieved state of the art predictive performances (Table 12). Its graph representation enables the extraction of critical account relationships, which is instrumental in addressing smart contract vulnerabilities.

Additionally, Fig. 3 also illustrates that out of the 76 vulnerabilities identified in (Zaazaa and Bakkali, 2023), less than a quarter were thoroughly addressed in the literature. Only three vulnerabilities: Reentrancy (SW-107), block timestamp/number manipulation (SW-116), and Integer Over/Underflow (SW-101), received in-depth scrutiny. This limitation is partly due to the representation of vulnerabilities available in the known datasets.

RQ2. What feature types and feature engineering methods are employed to address smart contract vulnerabilities in the Ethereum Blockchain?

Earlier studies typically used datasets with a single data type, such as opcodes, source code, or transactional data and only one method of feature extraction as observed in studies (Momeni et al., 2019) and (Wang et al., 2021). Over time, feature engineering has become increasingly advanced. In more recent research articles, authors have started to combine multiple data sources, such as opcodes, source code, and transactional data from the inspected smart contracts (e.g., (Liu et al., 2022) and (Deng et al., 2023). Data transformations have also

Table 14
Overview of the customised deep learning architectures employed for smart contract vulnerability detection along with corresponding github link (Reproducible column) whenever applicable. This table summarises the main approach used in each article of this section.

Paper	Main Innovation	SC Vulnerabilities	Machine Learning Approach	Sampling Technique	Reproducible	Achieved Performance
(Xu et al., 2023)	A TextCNN is trained using opcode sequences. These opcode sequences were vectorized with a Skip-Gram algorithm. To increase the model's resistance to redundant codes, random useless codes were inserted into the smart contracts.	12 types of Vulnerabilities	TextCNN	Balanced dataset	No	Best F1-Score: 97.24 % (SWE-101)
(Jie et al., 2021)	Fusion of intramodality and cross-modality feature vectors using spatial pyramid pooling and cross attention. The fused feature vectors are then processed through a bidirectional long short-term memory (Bi-LSTM) with a self-attention mechanism and a conditional random field (CRF) layer.	Binary Classification	TextCNN + RF (SC), GCN + CNN (BB), GCN (EVMB) . Then applied Bi-LSTM + CRF to those outputs	SMOTE	No	F1-score: 94.52 %
(Zhang et al., 2022)	An Ensemble Learning (EL) that combines multiple neural networks (NNs). The model follows a transfer knowledge methodology, initially pre-trained with a subset of the training dataset, followed by a fine-tuning using a different subset.	5 Vulnerabilities (SWE-[101, 107, 114, 116, 139])	**Assembled Model with:** CNN, RNN, RCNN, DNN, GRU, BiGRU, and Transformer	None. Imbalance Training	Yes (SCVDIE, 2024)	F1-Score: 97.42 %
(Zhang et al., 2023)	ReVulDL: Reentrancy Vulnerability Detection and Localization Tool. The vulnerability detection part consists of 6 parts: input units, join layer, masked multi-head attention layer, normalisation layers, n transformer layers and linear layers.	SWE-107	Customise Deep learning Model	None. Imbalance Training	Yes (IAContract, 2024)	F1-Score: 93.00 %
(Balcı et al., 2023)	A transformer model composed by self-attention mechanism and a position-wise fully connectedfeed-forward network.	Binary Classification	Transformer architecture	None. Imbalance Training	No	F1-Score: 86 %
(Gong et al., 2023)	Using a control flow graph based on smart contract opcodes to train a model based on transformers for smart contract vulnerability detection.	Binary Classification	Transformer architecture	None. Imbalance Training	No	F1-Score: 95.17 %
(Zuo et al., 2023)	A self-supervised learning (SSL) model, consisting of an encoder built using a Transformer architecture and a CNN and discriminator that is a similarity calculator, is trained on a discriminative task to enhance feature extraction.	Binary problem	Transformer Encoder and CNN	Balanced dataset	No	Precision: 87.60 %
(Yuan Y, Xie T. SVChecker: a deep learning-based system for smart contract vulnerability detection. In: Lu Y, Cheng C, editors. International Conference on Computer Application and Information Security (ICCAIS, 2021)	A method is proposed to extract specific code snippets from the source code, followed by feature extraction using Word2Vec, and then training a transformer neural network.	Binary Classification	Transformer architecture	Balanced dataset	Yes (Ye Yuan, 2021)	F1-Score: 94.82 %
(Deng et al., 2023)	Three NN classifiers, each utilising a distinct feature set derived from the representations of smart contract.	4 Vulnerabilities (SWE-[101, 107, 114, 138])	**3 NN Classifiers** from the 3 types of generated features.	ADASYN	No	Best ROC-AUC: 88.60 %

evolved, with extensive use of techniques like AST and CFG, specially for the GNN applications (e.g., (Nguyen et al, xxxx). Additionally, feature extraction methods such as n-grams, TF-IDF, and Word2Vec have become very common in this field of research.

RQ3. What are the key challenges and the next steps within smart contract vulnerabilities for the Ethereum blockchain?

In this comprehensive review, some studies demonstrate the integration of machine learning methods with fuzzy testing or expert rules. Ongoing exploration of these intersections is essential for developing frameworks capable of detecting vulnerabilities both before and after deployment. The growing research on Large Language Models (LLMs) for vulnerability detection, pinpointing specific parts of the code and auto-completing source code generation, contributes to deploying more secure smart contracts. Additionally, exploring avenues like active learning, reinforcement learning, and genetic programming could facilitate the automatic evolution of algorithms to detect emerging types of vulnerabilities. Another challenge will be the deploying of these ML models in real-time and at scale. Optimising these models for live blockchain environments to handle large transaction volumes efficiently should also be a focus of future research.

Table 15
Description of the dataset creation for the LLMs approaches, including the feature identification process, dataset size, labelling approach, and, when available, the link to the data .

Paper	Dataset Composition	Size	Labelling Approach	Dataset source link
(Ashizawa et al., 2021)	Abstract syntax tree, Assembly code, EVM Bytecode	5,000 Smart Contracts	Used the approach from (Momeni et al., 2019)	Etherscan
(Yan et al., 2022)	– Features from Abstract syntax tree (AST) – Source Code	99 Smart Contracts	Slither and Mythril	Own dataset, no source info.
(Duan et al., 2023)	– bi-gram from opcodes – Token generated from BERT model	**	Oyente, Slither andDefectChecker	Smart-Contract-Dataset (Smart, 2024)
(Sun et al., 2023)	Add the vocabulary from the official language description released by Ethereum to the vocabulary of the Bert model.	20,829 Smart Contracts (from known datasets)	Oyente and SmartCheck	Smartbugs (GitHub, 2023a) SolifiFI-benchmark (GitHub, 2023b) SoliAudit-benchmark (SoliAudit, 2024)

Source. (**) - not found

RQ4. What are the trade-offs between model complexity and performance?

In the context of smart contract vulnerability detection, more complex models, such as deep neural networks (e.g., CNNs, RNNs, GNNs) and custom architectures, are frequently explored for their capacity to capture nuanced features in code representations. However, our findings indicate several cases (Liu et al., 2022); (Balcı et al., 2023)) where increased model complexity did not yield significant performance gains compared to more straightforward approaches (Wang et al., 2021). This raises critical questions about the added value of complexity in this domain.

One potential explanation for this trend lies in the characteristics of the data itself. Many vulnerability datasets are imbalanced and limited in size, which can impede the effectiveness of complex models that require extensive training data to generalise well. Consequently, simpler models, particularly those leveraging robust feature engineering techniques (e.g., n-grams, AST-based representations), may achieve high accuracy without the risk of overfitting—a common challenge for complex models on smaller datasets.

In conclusion, while complex architectures may offer marginal improvements under certain conditions, simpler models often provide robust, interpretable results with lower computational costs.

Furthermore, the choice of performance metrics affects how the efficacy of model complexity is perceived. For imbalanced datasets, metrics like Precision-Recall AUC provide a more accurate assessment of model performance than accuracy or even standard ROC-AUC. Studies that did not report PR-AUC or F1-Score might have overestimated the benefits of model complexity, particularly if the chosen metrics did not fully capture the nuances of class imbalance.

5. Conclusion

While machine learning has advanced significantly in detecting vulnerabilities in smart contracts, a gap exists in the literature for an extensive review connecting similarities and discrepancies among machine learning approaches. To fill this gap, this paper innovatively explores machine learning applications in smart-contract security, providing valuable references for the community. It conducts an in-depth analysis and classification of machine-learning techniques for detecting vulnerabilities in smart contracts. The integration of advanced machine learning models becomes crucial for Ethereum's security as the network evolves, emphasising the need for continued research in this field.

We would also like to highlight a significant gap in the literature concerning the constraints and practical applications of the models discussed. While many articles propose innovative theoretical frameworks, only a limited number offer insights into the challenges encountered in real-world implementations. This underscores the need for future research to not only advance theoretical developments but also to critically evaluate their applicability and limitations in real-world scenarios.

Finally, this review contributes to fortifying the Ethereum blockchain against fraudulent activities, promoting trust, and enhancing security in decentralised systems.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Table 16
Overview of the LLM approaches employed for smart contract vulnerability detection along with corresponding github link (Reproducible column) whenever applicable. This table summarises the main approach used in each article of this section.

Paper	Main Innovation	SC Vulnerabilities	Machine Learning Approach	Sampling Technique	Reproducible	Achieved Performance
(Ashizawa et al., 2021)	Eth2Vec: – automated feature extraction leveraging natural language frameworks. –PV-DM method combined with similarity threshold to identify vulnerabilities.	6 Vulnerabilities (SWE-[100, 101, 107, 116, 134, 148])	Paragraph Vector – Distributed Memory (PV-DM) – also known as Doc2Vec.	Imbalance Training	Yes (GitHub, 2023g)	F1-Score: 57.5 %
(Yan et al., 2022)	Focusing on reentrancy vulnerability detection, an algorithm leveraging semantic-analysis is introduced here.	Reentrancy (SWE-107)	Semantic-analysis based methods	Imbalance Training	No	F1-Score: 93.89 %
(Duan et al., 2023)	Bi-gram features from opcodes combined with codeBERT tokens made the feature space used to train 7 ML methods.	3 Vulnerabilities (SWE-[107, 114, 116])	k-NN, SVM, CNN, RF, MLP,RNN and LSTM,	Balanced Dataset	No	Best Macro-F1 score: 93.30 %
(Sun et al., 2023)	Using Bert model with active learning and semi-supervised learning to effectively reduce the cost of labelling and improving the detection efficiency of the model.	6 Vulnerabilities (SWE –[101,107, 114, 115, 116, 139])	Bert model with active learning and semi-supervised learning.	Imbalance Training	No	Best F1-score: 67.30 %

Data availability

No data was used for the research described in the article.

References

Buterin V. A next-generation smart contract and decentralized application platform. finpedia.vn. Available: https://finpedia.vn/wp-content/uploads/2022/02/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf.

Coin Market Cap. In: CoinMarketCap [Internet]. [cited 14, Jan 2024]. Available: https://coinmarketcap.com/.

Zhang, W., & Anand, T. (2022). Ethereum Architecture and Overview. In W. Zhang, & T. Anand (Eds.), *Blockchain and Ethereum Smart Contract Solution Development: Dapp Programming with Solidity* (pp. 209–244). Berkeley, CA: Apress.

Dannen, C. (2017). *Introducing Ethereum and Solidity.*. https://doi.org/10.1007/978-1-4842-2535-6

Atzei, N., Bartoletti, M., & Cimoli, T. (2017). *A Survey of Attacks on Ethereum Smart Contracts (SoK)* (pp. 164–186). Principles of Security and Trust. Springer: Berlin Heidelberg.

Kehrli J. Blockchain 2.0-from bitcoin transactions to smart contract applications. Niceideas, November Available at: https://www niceideas ch/roller2/badtrash/entry/blockchain-2-0-frombitcoin (Accessed: 5 January 2018).

Lo, S. K., Liu, Y., Chia, S. Y., Xu, X., Lu, Q., Zhu, L., et al. (2019). Analysis of Blockchain Solutions for IoT: A Systematic Literature Review. *IEEE Access., 7*, 58822–58835.

Merlo, V., Pio, G., Giusto, F., & Bilancia, M. (2023). On the exploitation of the blockchain technology in the healthcare sector: A systematic review. *Expert Syst Appl., 213*, Article 118897.

Anoop, V. S., Asharaf, S., Goldston, J., & Williams, S. (2022). Blockchain for Industry 4.0: Blockchain for Industry 4.0: Emergence. *Challenges, and Opportunities. CRC Press*.

Wang, Z., & Guan, S. (2023). A blockchain-based traceable and secure data-sharing scheme. *PeerJ Comput Sci., 9*, e1337.

Jain, V. K., & Tripathi, M. (2023). An integrated deep learning model for Ethereum smart contract vulnerability detection. *Int J Inf Secur.*. https://doi.org/10.1007/s10207-023-00752-5

Hacks. In: Defillama [Internet]. [cited 4 Jan 2024]. Available: https://defillama.com/hacks.

Kushwaha, S. S., Joshi, S., Singh, D., Kaur, M., & Lee, H.-N. (2022). Ethereum Smart Contract Analysis Tools: A Systematic Review. *IEEE Access., 10*, 57037–57062.

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., et al. (2021). The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *Int J Surg., 88*, Article 105906.

Website. Available: Review-on-ML-Methods-for-Detecting-Smart-Contracts-Vulnerabilities-within-Ethereum-Blockchain." Github. June 18, 2024. https://github.com/joaocrisostomo/Review-on-ML-methods-for-detecting-Smart-Contracts-vulnerabilities-within-Ethereum-Blockchain.

Kushwaha, S. S., Joshi, S., Singh, D., Kaur, M., & Lee, H.-N. (2022). Systematic Review of Security Vulnerabilities in Ethereum Blockchain Smart Contract. *IEEE Access., 10*, 6605–6621.

Hou W, Cui B, Li R. A Survey on Blockchain Data Analysis. 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC). IEEE; 2021. pp. 357–365.

Hisham, S., Makhtar, M., & Aziz, A. A. (2022). Combining multiple classifiers using ensemble method for anomaly detection in blockchain networks: A comprehensive review. *Int J Adv Comput Sci Appl., 13*. https://doi.org/10.14569/ijacsa.2022.0130848

Jiang F, Chao K, Xiao J, Liu Q, Gu K, Wu J, et al. Enhancing Smart-Contract Security through Machine Learning: A Survey of Approaches and Techniques. Electronics.

Li, Z., Lu, S., Zhang, R., Zhao, Z., Liang, R., Xue, R., et al. (2023). VulHunter: Hunting Vulnerable Smart Contracts at EVM Bytecode-Level via Multiple Instance Learning. *IEEE Trans Software Eng., 49*, 4886–4916.

Nguyen HH, Nguyen N-M, Xie C, Ahmadi Z. MANDO-HGT: Heterogeneous Graph Transformers for Smart Contract Vulnerability Detection.

Mahesh, B. (2020). Machine learning algorithms-a review. Available: *International Journal of Science and Research (IJSR)* https://www.researchgate.net/profile/Batta-Mahesh/publication/344717762_Machine_Learning_Algorithms_-A_Review/links/5f8b2365299bf1b53e2d243a/Machine-Learning-Algorithms-A-Review.pdf?eid=5082902844932096.

Shrestha, A., & Mahmood, A. (2019). Review of Deep Learning Algorithms and Architectures. *IEEE Access., 7*, 53040–53065.

Zaazaa, O., & Bakkali, H. E. (2023). Unveiling the landscape of smart contract vulnerabilities: A detailed examination and codification of vulnerabilities in prominent Blockchains. *Int J Comput Netw Commun., 15*, 55–75.

Momeni P, Wang Y, Samavi R. Machine Learning Model for Smart Contracts Security Analysis. 2019 17th International Conference on Privacy, Security and Trust (PST), Fredericton, NB, Canada, 2019, pp 1-6, doi: 101109/PST4712120198949045. 06-Jan-2020.

Liao J-W, Tsai T-T, He C-K, Tien C-W. SoliAudit: Smart Contract Vulnerability Assessment Based on Machine Learning and Fuzz Testing.

Xu, Y., Hu, G., You, L., & Cao, C. (2021). A Novel Machine Learning-Based Analysis Model for Smart Contract Vulnerability. Security and Communication. *Networks., 2021*. https://doi.org/10.1155/2021/5798033

Mandloi, J., & Bansal, P. (2022). A machine learning-based dynamic method for detecting vulnerabilities in smart contracts. *International Journal of Applied Engineering & Technology., 4*, 110–118.

J, Lohith J., K, Anusree Manoj, P, Guru Nanma, Srinivasan P. TP-Detect: trigram-pixel based vulnerability detection for Ethereum smart contracts. Multimed Tools Appl. 2023;82: 36379–36393.

Mezina, A., & Ometov, A. (2023). Detecting Smart Contract Vulnerabilities with Combined Binary and Multiclass Classification. *Cryptogr Commun., 7*, 34.

Goldberg Y, Levy O. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. ArXiv. 2014;abs/1402.3722. Available: http://arxiv.org/abs/1402.3722.

GitHub - smartbugs/smartbugs: SmartBugs: A Framework to Analyze Ethereum Smart Contracts. In: GitHub [Internet]. [cited 10 Dec 2023]. Available: https://github.com/smartbugs/smartbugs.

GitHub - DependableSystemsLab/SolidiFI-benchmark: Repository of benchmarks to evaluate Solidity Smart contract analysis tools. In: GitHub [Internet]. [cited 10 Dec 2023]. Available: https://github.com/DependableSystemsLab/SolidiFI-benchmark.

GitHub - smartbugs/smartbugs-wild: This repository contains 47,398 smart contracts extracted from the Ethereum network. In: GitHub [Internet]. [cited 10 Dec 2023]. Available: https://github.com/smartbugs/smartbugs-wild.

Securify2. In: https://github.com/ [Internet]. [cited 04, Jan 2024]. Available: https://github.com/eth-sri/securify2.

Bigquery Ethereum-Blockchain. In: https://www.kaggle.com/ [Internet]. [cited January, 19 2024]. Available: https://www.kaggle.com/datasets/bigquery/ethereum-blockchain.

SoliAudit. In: https://github.com/ [Internet]. [cited 19, January 2024]. Available: https://github.com/jianwei76/SoliAudit.

Wang W, Song J, Xu G, Li Y, Wang H, Su C. ContractWard: Automated Vulnerability Detection Models for Ethereum Smart Contracts. IEEE Transactions on Network Science and Engineering. 01 April-June 2021;8: 1133–1144.

Shakya, S., Mukherjee, A., Halder, R., Maiti, A., & SmartMixModel, C. A. (2022). Machine learning-based vulnerability detection of solidity smart contracts. In *2022 IEEE International Conference on Blockchain (Blockchain). IEEE*. https://doi.org/10.1109/blockchain55522.2022.00016

Yang H, Zhang J, Gu X, Cui Z. Smart Contract Vulnerability Detection based on Abstract Syntax Tree. 2022 8th International Symposium on System Security, Safety, and Reliability (ISSSR). IEEE; 2022. pp. 169–170.

Aljofey A, Rasool A, Jiang Q, Qu Q. A Feature-Based Robust Method for Abnormal Contracts Detection in Ethereum Blockchain. doi:10.1007/BF00058655.

J J L, Singh K, Chakravarthi B. Digital forensic framework for smart contract vulnerabilities using ensemble models. Multimed Tools Appl. 2023. doi:10.1007/s11042-023-17308-3.

Eshghie, M., Artho, C., & Gurov, D. (2021). In *Dynamic Vulnerability Detection on Smart Contracts Using Machine Learning* (pp. 305–312). New York, NY, USA: Association for Computing Machinery.

Xue, Y., Ye, J., Zhang, W., Sun, J., Ma, L., Wang, H., et al. (2022). xFuzz: Machine Learning Guided Cross-Contract Fuzzing. *IEEE Trans Dependable Secure Comput., PP*, 1–14.

Sathiyamurthy, L. K. A. (2022). Towards Auto Contract Generation and Ensemble-based Smart Contract Vulnerability Detection. *International Journal of Electrical and Computer Engineering Systems., 13*(9).

GitHub - tintinweb/smart-contract-sanctuary-ethereum: A home for ethereum smart contracts. In: GitHub [Internet]. [cited 10 Dec 2023]. Available: https://github.com/tintinweb/smart-contract-sanctuary-ethereum.

GitHub - abdul-rasool/Abnormal-Contracts-Detection: A Feature-Based Robust Method for Abnormal Contracts Detection in Ethereum Blockchain. In: GitHub [Internet]. [cited 10 Dec 2023]. Available: https://github.com/abdul-rasool/Abnormal-Contracts-Detection.

GitHub - pedrocrvz/smartcheck: SmartCheck – a static analysis tool that detects vulnerabilities and bugs in Solidity programs (Ethereum-based smart contracts). In: GitHub [Internet]. [cited 10 Dec 2023]. Available: https://github.com/pedrocrvz/smartcheck.

ContractWard. In: https://github.com/ [Internet]. [cited 19, January 2024]. Available: https://github.com/Dattu219/ContractWard.

Hu, T., Liu, X., Chen, T., Zhang, X., Huang, X., Niu, W., et al. (2021). Transaction-based classification and detection approach for Ethereum smart contract. *Inf Process Manag., 58*, Article 102462.

Goswami S, Singh R, Saikia N, Bora KK, Sharma U. TokenCheck: Towards Deep Learning Based Security Vulnerability Detection In ERC-20 Tokens. 2021 IEEE Region 10 Symposium (TENSYMP). IEEE; 2021. pp. 1–8.

Zhu, J., Xing, X., Wang, G., & Li, P. (2023). Opcode sequences-based smart contract vulnerabilities detection using deep learning. In *2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). IEEE*. https://doi.org/10.1109/trustcom60117.2023.00057

Gopali S, Khan ZA, Chhetri B, Karki B, Namin AS. Vulnerability detection in smart contracts using deep learning. 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC). IEEE; 2022. doi:10.1109/compsac54236.2022.00197.

Zhang, X., Li, J., & Wang, X. (2022). Smart Contract Vulnerability Detection Method based on Bi-LSTM Neural Network. In *2022 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA). IEEE* (pp. 38–41).

Demir HO, Parlat SZ, Gumus A. Ethereum Blockchain Smart Contract Vulnerability Detection Using Deep Learning. 2023 7th International Symposium on Innovative Approaches in Smart Technologies (ISAS). IEEE; 2023. pp. 1–5.

Jain, V. K., & Tripathi, M. (2023). Multi-Objective Approach for Detecting Vulnerabilities in Ethereum Smart Contracts. In *2023 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC). IEEE* (pp. 1–6).

Tereshchenko O, Komleva N. Vulnerability Detection of Smart Contracts Based on Bidirectional GRU and Attention Mechanism. Information and Communication Technologies in Education, Research, and Industrial Applications. Springer Nature Switzerland; 2023. pp. 276–287.

Google Cloud Platform. [cited 10 Dec 2023]. Available: https://console.cloud.google.com/bigquery?project=onyx-cosmos-300017&ws=!1m4!1m3!3m2!1sbigquery-public-data!2sethereum_blockchain.

VulHunter. In: https://github.com/ [Internet]. [cited 04, Jan 2024]. Available: https://github.com/Secbrain/VulHunter/tree/main.

Xing, C., Chen, Z., Chen, L., Guo, X., Zheng, Z., & Li, J. (2020). A new scheme of vulnerability analysis in smart contract with machine learning. *Wireless Networks.*. https://doi.org/10.1007/s11276-020-02379-z

Sun Y, Gu L. Attention-based Machine Learning Model for Smart Contract Vulnerability Detection. 2021. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1820/1/012004.

Zhou, Q., Zheng, K., Zhang, K., Hou, L., & Wang, X. (2022). Vulnerability Analysis of Smart Contract for Blockchain-Based IoT Applications: A Machine Learning Approach. *IEEE Internet of Things Journal., 9*, 24695–24707.

Zhang, L., Wang, J., Wang, W., Jin, Z., Su, Y., & Chen, H. (2022). Smart contract vulnerability detection combined with multi-objective detection. *Computer Networks., 217*, Article 109289.

Wu, Z., Li, S., Wang, B., Liu, T., Zhu, Y., Zhu, C., et al. (2022). Detecting vulnerabilities in ethereum smart contracts with deep learning. In *2022 4th International Conference on Data Intelligence and Security (ICDIS). IEEE*. https://doi.org/10.1109/icdis55630.2022.00016

He D, Ding K, Chan S, Guizani M. Unknown Threats Detection Methods of Smart Contracts. IEEE Internet of Things Journal. PP: 1–1.

Finn C, Abbeel P, Levine S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In: Precup D, Teh YW, editors. Proceedings of the 34th International Conference on Machine Learning. PMLR; 06–11 Aug 2017. pp. 1126–1135.

Yang, Z., Zhu, W., & Yu, M. (2023). Improvement and Optimization of Vulnerability Detection Methods for Ethernet Smart Contracts. *IEEE Access., 11*, 78207–78223.

MODNN - Multiple-Objective Detection Neural Network. In: https://github.com/ [Internet]. [cited 06, Jan 2024]. Available: https://github.com/yangzuwjl/MODNN.

ScrawID. In: https://github.com [Internet]. [cited 07, Jan 2024]. Available: https://github.com/sujeetc/ScrawID.

Ashizawa N, Yanai N, Cruz JP, Okamura S. Eth2Vec: Learning Contract-Wide Code Representations for Vulnerability Detection on Ethereum Smart Contracts. Proceedings of the 3rd ACM International Symposium on Blockchain and Secure Critical Infrastructure. New York, NY, USA: Association for Computing Machinery; 2021. pp. 47–59.

Zkik, K., Sebbar, A., Fadi, O., Mustapha, O., & Belhadi, A. (2023). In *A graph neural network approach for detecting smart contract anomalies in collaborative economy platforms based on blockchain technology*. Decision and Information Technologies (CoDIT). IEEE. https://doi.org/10.1109/codit58514.2023.10284080.

Liu, L., Tsai, W.-T., Bhuiyan, M. Z. A., Peng, H., & Liu, M. (2022). Blockchain-enabled fraud discovery through abnormal smart contract detection on Ethereum. *Future Gener Comput Syst., 128*, 158–166.

Zeng, Q., He, J., Zhao, G., Li, S., Yang, J., Tang, H., et al. (2022). EtherGIS: A Vulnerability Detection Framework for Ethereum Smart Contracts Based on Graph Learning Features. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)* (pp. 1742–1749).

Liu, Z., Jiang, M., Zhang, S., Zhang, J., & Liu, Y. (2023). A Smart Contract Vulnerability Detection Mechanism Based on Deep Learning and Expert Rules. *IEEE Access., 11*, 77990–77999.

Nguyen HH, Nguyen N-M, Xie C, Ahmadi Z, Kudendo D, Doan T-N, et al. MANDO: Multi-Level Heterogeneous Graph Embeddings for Fine-Grained Detection of Smart Contract Vulnerabilities. arXiv [cs.SE]. 2022. Available: http://arxiv.org/abs/2208.13252.

Nguyen, H. H., Nguyen, N.-M., Doan, H.-P., Ahmadi, Z., Doan, T.-N., & Jiang, L. (2022). In *MANDO-GURU: vulnerability detection for smart contract source code by heterogeneous graph embeddings* (pp. 1736–1740). New York, NY, USA: Association for Computing Machinery.

Hobor A, Abdelaziz T. Smart Learning to Find Dumb Contracts. The 32nd USENIX Security Symposium (USENIX 2023). 2023.

Wang, Z., Zheng, Q., & Sun, Y. (2022). In *GVD-net: Graph embedding-based machine learning model for smart contract vulnerability detection*. and Information Technology (ADMIT). IEEE. https://doi.org/10.1109/admit57209.2022.00024.

Bartoletti, M., Carta, S., Cimoli, T., & Saia, R. (2020). Dissecting Ponzi schemes on Ethereum: Identification, analysis, and impact. *Future Gener Comput Syst., 102*, 259–277.

VNT Chain is an experimental public blockchain platform proposed by companies and universities from Singapore, China, and Australia. Vntchain smart contacts. In: Google Drive [Internet]. 9 Feb 2020 [cited 20 Jun 2024]. Available: https://drive.google.com/drive/folders/1FTb_ERCOGNGM9dTeHLwAxBLw7X5Td4v.

Ethereum Dataset Large. In: Google Sheets [Internet]. [cited 21, Jan 2024]. Available: https://bit.ly/EthereumSC_Dataset_Large.

EthereumSC Dataset Small. In: Google Sheets [Internet]. [cited 21, Jan 2024]. Available: https://bit.ly/EthereumSC_Dataset_Small.

MANDO-Project: ge-sc-Transformer. In: Github [Internet]. [cited 16, Jan 2024]. Available: https://github.com/MANDO-Project/ge-sc-transformer.

MANDO-Project: ge-sc. In: Github [Internet]. [cited 21, Jan 2024]. Available: https://github.com/MANDO-Project/ge-sc.

DLVA-Tool. In: Google Drive [Internet]. [cited 21, Jan 2024]. Available: https://bit.ly/DLVA-Tool.

Xu, Z., Chen, X., Dong, X., Han, H., Yan, Z., Ye, K., et al. (2023). An efficient code-embedding-based vulnerability detection model for Ethereum smart contracts. *Int J Data Warehouse Min., 19*, 1–23.

Jie, W., Koe, A. S. V., Huang, P., & Zhang, S. (2021). Full-Stack Hierarchical Fusion of Static Features for Smart Contracts Vulnerability Detection. In *2021 IEEE International Conference on Blockchain (Blockchain). IEEE* (pp. 95–102).

Deng, W., Wei, H., Huang, T., Cao, C., Peng, Y., & Hu, X. (2023). Smart Contract Vulnerability Detection Based on Deep Learning and Multimodal Decision Fusion. *Sensors ., 23*. https://doi.org/10.3390/s23167246

Zhang, L., Wang, J., Wang, W., Jin, Z., Zhao, C., Cai, Z., et al. (2022). A Novel Smart Contract Vulnerability Detection Method Based on Information Graph and Ensemble Learning. *Sensors ., 22*. https://doi.org/10.3390/s22093581

Zhang, Z., Lei, Y., Yan, M., Yu, Y., Chen, J., Wang, S., et al. (2023). In *Reentrancy Vulnerability Detection and Localization: A Deep Learning Based Two-phase Approach* (pp. 1–13). New York, NY, USA: Association for Computing Machinery.

Balcı, E., Yılmaz, G., Uzunoğlu, A., & Soyak, E. G. (2023). Accelerating smart contract vulnerability scan using transformers. In *2023 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)* (pp. 1–6).

Gong, K., Song, X., Wang, N., Wang, C., & Zhu, H. (2023). SCGformer: Smart contract vulnerability detection based on control flow graph and transformer. *IET Blockchain., 3*, 213–221.

Zuo, H., Shi, Y., Qin, Z., & Jiang, X. (2023). Smart Contract Bytecode Similarity Detection Based on Self-supervised Learning. In *2023 8th International Conference on Signal and Image Processing (ICSIP). IEEE* (pp. 804–808).

Yuan Y, Xie T. SVChecker: a deep learning-based system for smart contract vulnerability detection. In: Lu Y, Cheng C, editors. International Conference on Computer Application and Information Security (ICCAIS 2021). SPIE; 2022. doi:10.1117/12.2637775.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. Available: *Int Conf Learn Represent.* http://arxiv.org/abs/1301.3781.

Website. Available: starStraw. 2022. "OpcodeToVec." Github. June 29, 2022. https://github.com/starStraw/OpcodeToVec.

GitHub - fseclab-osaka/eth2vec. In: GitHub [Internet]. [cited 10 Dec 2023]. Available: https://github.com/fseclab-osaka/eth2vec.

SCVDIE. In: Github [Internet]. [cited 12, Jan 2024]. Available: https://github.com/yzu-wjl/SCVDIE.

IAContract. In: Github [Internet]. [cited 14, Jan 2024]. Available: https://github.com/toolstemp/IAcontract.

Ye Yuan TX. SVChecker. In: Github [Internet]. 28 Nov 2021 [cited 20 Jun 2024]. Available: https://github.com/yesmola/SVChecker.

Jin W, He T, Qian Y, Yu K. Paragraph vector based topic model for language model adaptation. Interspeech 2015. ISCA: ISCA; 2015. doi:10.21437/interspeech.2015-697.

Duan, L., Yang, L., Liu, C., Ni, W., & Wang, W. (2023). A New Smart Contract Anomaly Detection Method by Fusing Opcode and Source Code Features for Blockchain Services. *IEEE Trans Netw Serv Manage., 20*, 4354–4368.

Sun, X., Tu, L., Zhang, J., Cai, J., Li, B., & Wang, Y. (2023). ASSBert: Active and semi-supervised bert for smart contract vulnerability detection. *Journal of Information Security and Applications., 73*, Article 103423.

Storhaug A, Li J, Hu T. Efficient Avoidance of Vulnerabilities in Auto-completed Smart Contract Code Using Vulnerability-constrained Decoding. 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE). IEEE; 2023. pp. 683–693.

Mathur N, Baldwin T, Cohn T. Tangled up in BLEU: Reevaluating the Evaluation of Automatic Machine Translation Evaluation Metrics. arXiv [cs.CL]. 2020. Available: http://arxiv.org/abs/2006.06264.

Yan, X., Wang, S., & Gai, K. (2022). A Semantic Analysis-Based Method for Smart Contract Vulnerability. In *2022 IEEE 8th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS). IEEE* (pp. 23–28).

Smart-Contract-Dataset. In: Github [Internet]. [cited 22, Jan 2024]. Available: https://github.com/Messi-Q/Smart-Contract-Dataset.