A Survey on Trends and Challenges in AI-Powered Smart Contract Analysis

Anmol Mogalayi, Sridhar K S, Dr. Pijush Barthakur

Dept. of Master of Computer Applications, K.L.S Gogte Institute of Technology Belagavi, Karnataka, India

DOI: https://doi.org/10.51584/IJRIAS.2025.100700101

Received: 08 July 2025; Accepted: 15 July 2025; Published: 15 August 2025

ABSTRACT

Smart contracts are the backbone of decentralized applications, enabling secure and autonomous execution of digital contracts on blockchain platforms. However, increasing complexity and immutability of these contracts are causing severe security threats. Traditional auditing techniques, although helpful, are limited in scalability and mostly incapable of detecting emerging vulnerabilities. This has led to a growing desire to extend the deployment of Artificial Intelligence (AI) and Natural Language Processing (NLP) techniques to the audit and security of smart contracts. In this paper, we present a comprehensive review of AI-based approaches of auditing and securing smart contracts, highlighting recent advances in machine learning, deep learning, and transformer-based architectures. We discuss stateof-the-art tools and frameworks, compare their methodology, and outline their respective strengths and weaknesses. We also discuss important challenges like availability of datasets, generalization to unknown vulnerabilities, interpretability of AI results, and integration with existing blockchain platforms. Finally, we discuss future research directions and propose future work for the development of more secure, intelligent, and scalable systems for analyzing smart contracts.

Keywords: Smart Contracts, Vulnerability Detection, Artificial Intelligence, Natural Language Processing, Blockchain Security

INTRODUCTION

Smart contracts are a core component of blockchain application, especially on platforms such as Ethereum. They are bits of computer programming that activate automatically upon a set of conditions being satisfied, without the necessity of a middleman in digital contracts. Their ability to automate transactions and enforce rules has propelled rapid adoption across industries. The same features that provide power to smart contracts immutability, transparency, and autonomous execution also pose enormous security threats. Once put in place, a contract is immutable, transparency, and autonomous execution also introduce significant security challenges.

Several high-profile attacks, including The DAO hack [1], the Parity wallet bug [2], and a spate of recent hacks against the DeFi protocols [3], have revealed deep vulnerabilities in smart contract code. Not only did the attacks result in financial losses, but they also questioned the security and integrity of blockchain systems. Thus, the need for trustworthy smart contract auditing has increased exponentially.

Traditional auditing techniques, including manual code examination and rule-based static analysis tools like Mythril [1], Oyente[2], and Slither[4], do offer security to some degree but are inherently restrictive. Manual verification takes enormous man-hours, is prone to human error, and cannot handle the growing number of contracts in the space efficiently. Static analyzers do have automation associated with them but are reliant on known signatures for vulnerabilities and cannot detect complex or emerging logical bugs. To overcome these shortcomings, researchers have moved towards Artificial Intelligence (AI) and Natural Language Processing (NLP) methods. These methods approach smart contract code as a formal language, where machine learning models can be applied for pattern detection, anomaly detection, and semantic analysis. Transformer models such as CodeBERT[4], GPT[5], and LLaMA[6] have been found to possess unprecedented abilities in source code analysis, and their use in smart contract auditing is being researched actively. This paper presents an in-depth review of AI-powered smart contract analysis, with emphasis on the convergence of blockchain security, AI, and NLP. We seek to classify and contrast existing efforts, find common approaches, point out their benefits and limitations, and specify the most outstanding challenges yet to be addressed. Based on this work, we also propose directions for future work that can make smart contract auditing systems more secure, understandable, and scalable.

Over the past few years, the speedy growth of decentralized finance (DeFi), tokenized assets, and Web3 platforms has tremendously accelerated the use of smart contracts in a big way. Programmable contracts are being used in a variety of industries such as insurance, gaming, supply chain, identity verification, and trading of digital assets. With the developing blockchain ecosystem, smart contracts have moved beyond their early uses in simple token trade or crowdfunding and now support complex logic and high-value transactions worth billions of dollars.

But this added complexity has turned smart contracts into a target-rich environment for malicious behavior. Their deterministic nature, once on-chain, is cemented in place—so that even small errors in coding can have irreversible consequences. As smart contracts handle increasingly complex logic and high-value operations, even minor defects in their code can escalate into widespread security vulnerabilities.

Classic security practices in software development have generally included iterative patching and updates. Smart contracts, however, are resistant by nature to post-deployment changes. This aspect increases the focus on strict pre-deployment security testing and accurate vulnerability detection.

LITERATURE REVIEW

The area of smart contract security has progressed considerably, with researchers and practitioners exploring various automated methods to discover vulnerabilities. The work initially started with rule-based static analysis tools such as Mythril, Oyente, and Slither [1][2][4]. They execute bytecode or contract source code symbolically and by pattern matching to identify known types of vulnerabilities such as integer overflows, access control issues, unchecked call returns, and timestamp dependencies, integer overflow, and unchecked return value call. However, their pre-existing rule dependency limits them from identifying more complex or newer bugs.

To counteract the drawbacks of traditional tools, machine learning (ML) techniques have been used. For instance, Liu et al. used deep neural networks to classify Solidity smart contracts based on their vulnerability features [3]. Similarly, Tereshchenko and Komleva introduced a methodology that utilizes attention-based networks like CodeBERT to model the semantic and syntactic features of smart contract code, leading to higher detection rates for annotated datasets [4].

Recent research has evolved from simple classifiers towards more complex frameworks. Khodadadi and Tahmoresnezhad created a multimodal hybrid model merging token embeddings and control flow graph (CFG) features with FastText and BiGRU. Their approach showed solid performance on well-known benchmark sets like ScrawlD [5]. Mi et al. further proposed a metric-learning approach that scans low-level bytecode representations with neural embeddings for detecting subtle security flaws, especially those that might not be apparent at the high-level code [6]. Transformer models have attracted more attention in this area due to their promise.

Models such as CodeBERT, GraphCodeBERT, and finetuned GPT and LLaMA variants have demonstrated their promise in vulnerability detection, code summarization, and audit explanation [7][8]. Yu (2024) put forward a Retrieval-Augmented Generation (RAG) system based on vector search and large language models (LLMs) to improve the accuracy and explainability of smart contract analysis [9]. Following the same line, Wei et al. introduced LLM-SmartAudit, a multiagent system where multiple instances of GPT work together to simulate human-like auditing activities [10].

Several in-depth reviews have tried to profile the area of AI-based smart contract analysis. De Baets et al. reviewed over 80 papers, categorizing approaches into static, dynamic, and hybrid, and also pointing to the growing relevance of Graph Neural Networks (GNNs) and transformers [11]. Ozdag (2025) compiled top shortcomings in current datasets, such as small sizes and poor label consistency, and also pointed to the relevance of explainable artificial intelligence (XAI) for this safety-critical use case [12].

Despite the progress made, existing research suggests significant challenges. Among these challenges are the generalizability of models across diverse codebases, the lack of consistent evaluation standards, the difficulty of annotating large datasets, and the integration of artificial intelligence models into realistic blockchain development environments.

This study is based on previous reviews in that it not only addresses the technical methods but also the overarching trends that govern this field, namely explainability, scalability, and applicability to the real world. We classify the reviewed methods in the following section and identify the most significant methodologies in the analysis of artificial intelligence-based smart contracts.

Dataset Landscape and Benchmarking Gaps

The development of AI-based smart contract auditing is directly tied to the availability and reliability of data sets on which it is trained, tested, and benchmarked. Yet the present data ecosystem of freely available data sets is severely constrained in size and representativeness.

Most of the literature depends on datasets like SmartBugs, ScrawlD, or synthetically created sets of Solidity contracts. These datasets are prone to severe class imbalances, with vulnerabilities like integer overflow appearing overly frequent, while less evident or logic-based ones are rare. Moreover, several of these datasets comprise rather short, artificial examples that do not properly represent the sophistication of actual contracts used in decentralized finance (DeFi) or enterprise contexts.

In addition, there is inconsistency in the manner datasets are labeled and annotated. Datasets are labeled at either the function level or have line-level annotation or binary vulnerability flags. One of the largest obstacles is the lack of standardized benchmarking protocols. Different studies employ a variety of different metrics—accuracy, F1-score, precision, and recall—without a shared evaluation protocol. In addition, the absence of a shared dataset or challenge set impedes the making of consistent and reproducible comparisons between models. Developing shared benchmarks with uniform labeling, vulnerability taxonomies, and evaluation metrics is needed in order to enable significant progress and cross-study validation within this area.

Table I. Comparison of Selected Ai-Based Smart Contract Auditing Approaches

Authors	Dataset	Methodology	Contribution Summary
Liu et al. (2022)	SmartBugs	DNN-based token embeddings	Early ML classification for Solidity vulnerabilities
Khodadadi & Tahmoresnezhad (2023)	ScrawlD	BiGRU + FastText + CFG	Multimodal hybrid model combining token & structure
Mi et al. (2022)	Bytecode Set	Metric learning on bytecode	Detected low-level flaws using neural embeddings
Yu (2024)	Custom	GPT-4 + Vector Retrieval	Used RAG for semantic analysis & interpretability
Wei et al. (2024)	Open-source	Multi-agent GPT Auditing	Simulated expert audits using collaborative agents

In-Depth Review of Selected Research Works

The area of smart contract security has progressed considerably, with researchers and practitioners exploring various automated methods to discover vulnerabilities. The work initially started with rule-based static analysis tools such as Mythril, Oyente, and Slither. These tools symbolically execute bytecode or contract source code and perform pattern matching to identify known vulnerabilities such as reentrancy, integer overflow, and unchecked return values [1], [2], [3]. However, their reliance on predefined rules restricts their ability to detect newer or more complex vulnerabilities.

To overcome the limitations of traditional tools, machine learning (ML) techniques have been applied. Liu et al. [4] introduced deep neural networks to classify smart contracts based on vulnerability features. Tereshchenko and Komleva [5] proposed an attention-based model using CodeBERT to capture semantic and syntactic features, achieving higher detection rates.

Recent works have advanced beyond simple classifiers. Khodadadi and Tahmoresnezhad [6] proposed a hybrid multimodal model that merges token embeddings and control flow graphs using FastText and BiGRU, yielding strong results on datasets like ScrawlD. Mi et al. [7] introduced a metric-learning approach that uses low-level bytecode representations with neural embeddings to detect subtle security issues. Transformer models have emerged as promising tools. CodeBERT, GraphCodeBERT, and fine-tuned versions of GPT and LLaMA have shown success in vulnerability detection, summarization, and audit explainability [8], [9]. Yu [10] introduced a Retrieval-Augmented Generation (RAG) system that enhances accuracy using vector search and large language models. Wei et al. [11] proposed LLM-SmartAudit, a multi-agent GPT-based system simulating human audit behavior. In terms of surveys, De Baets et al. [12] reviewed over 80 papers and categorized them into static, dynamic, and hybrid approaches, emphasizing the growing relevance of graph neural networks (GNNs) and transformers. Ozdag [13] discussed key issues in datasets, such as limited size and inconsistent labeling, and stressed the need for explainable AI (XAI). Despite advances, challenges persist. These include generalization across diverse codebases, absence of consistent benchmarks, annotation difficulties, and limited integration of AI into real-world blockchain environments [14]. Our study builds on these reviews and contributes by outlining the most influential methodologies and identifying the prevailing trends in scalability, explainability, and real-world applicability.

METHODOLOGIES

AI-powered smart contract analysis has evolved across several distinct paradigms. Based on our literature review, these approaches can be broadly categorized into four methodological groups:

1. Rule-Based and Static Analysis Enhanced by AI
2. Supervised Learning with Code Embeddings
3. Transformer-Based Language Models
4. Hybrid and Multi-Modal Architectures

Each of these categories reflects a specific way of integrating AI and/or NLP into smart contract auditing workflows. Below, we explore each class in detail.

Rule-Based and Static Analysis Enhanced by AI

Traditional static analyzers such as Slither, Mythril, and Oyente operate on symbolic execution and heuristic rules. Recent studies have attempted to enhance these with AI components that can rank results, reduce false positives, or suggest fixes.

For example, Vulpedia abstracts patterns from known vulnerabilities and uses these to construct AI-generated vulnerability signatures. These rule-based approaches remain interpretable but lack generalizability, particularly for zeroday vulnerabilities or unconventional coding patterns.

Supervised Learning with Code Embeddings

Supervised models typically require labeled datasets where smart contracts are tagged with specific vulnerabilities. These models, like Bi-LSTM, CNNs, or GRUs, rely on embedding techniques such as FastText, word2vec, or Doc2Vec to convert source code into numerical vectors.

Khodadadi & Tahmoresnezhad developed a BiGRU classifier using tokenized source code, achieving high recall but limited interpretability. Models trained this way perform well on known datasets but struggle with new or adversarial examples due to data bias.

Transformer-Based Language Models

Recent progress in NLP has inspired the application of transformer-based models like CodeBERT, GPT-3/4, LLaMA, and GraphCodeBERT. These models treat code as a form of natural language, allowing them to learn syntax and semantics simultaneously.

Yu (2024) proposed a Retrieval-Augmented Generation (RAG) model where relevant code contexts are retrieved using vector similarity before being analysed by GPT-4. This hybrid greatly improves explainability and performance, particularly in ambiguous or lengthy contracts.

Another notable example, LLM-SmartAudit, coordinates multiple LLM agents in a cooperative setting—mimicking how multiple auditors might review the same code.

Hybrid and Multi-Modal Architectures

These models integrate various representations of code, such as ASTs (Abstract Syntax Trees), CFGs (Control Flow Graphs), and bytecode along with source-level text. For instance, Mi et al. used a metric-learning model that combines both symbolic and neural features. Others leverage GNNs and CNNs over CFGs and call graphs to capture deep structural patterns.

Hybrid methods are typically more robust and can balance performance with interpretability but require complex architecture and high computational resources.

Proposed Conceptual Framework

To unify the strengths of existing approaches, we propose a conceptual layered framework for AI-powered smart contract auditing and to combine the strongest features of current methods and prevent their weaknesses, we suggest modular and layered AI-enabled smart contract auditing architecture. The conceptual framework to be envisioned is meant to be extensible across various analysis engines, future-proof against emerging AI developments, and deployable and understandable in reality. The architecture consists of four main layers:

Preprocessing Layer

This bottom layer is responsible for pre-processing the smart contract code for analysis. It is composed of a number of extraction and transformation steps:

1. Source Code Retrieval: Source code written in Solidity or Vyper is retrieved from repositories or blockchain explorers.
2. Bytecode Extraction: In deployed contracts, Ethereum Virtual Machine (EVM) bytecode is extracted to enable lower-level analysis.
3. Abstract Syntax Tree (AST) Generation: ASTs describe the program syntax in a hierarchical, tree-like structure important for syntactic feature extraction.
4. Control Flow Graph (CFG) Construction: CFGs are the representation of execution flow through the contract to help comprehend logic paths and attack surfaces.

Such dense representations support reasoning from many viewpoints and structure data for symbolic and neural processing.

Analytical Framework

This layer performs the basic computational operations on the representations generated in the preprocessing stage. It possesses a hybrid analytical approach that combines both traditional symbolic techniques and AI models:

1. Symbolic Analysis Tools (e.g., Mythril, Slither): Perform rule-based detection of known patterns such as reentrancy or integer overflows.
2. Transformer-Based Language Models (e.g., CodeBERT, GPT): Operate on source code and bytecode embeddings to capture both syntax and semantics. These models are fine-tuned on security-specific tasks such as vulnerability classification, code summarization, and anomaly detection.

By fusing symbolic and neural techniques, the analysis layer provides an enhanced and more precise detection capability, such as the capability to detect zero-day vulnerabilities.

Reasoning Layer

The reasoning layer integrates higher-order logic and improves interpretability across the auditing process by:

1. LLM Coordination: Several instances of large language models (LLMs) mimic collaborative auditing through cross-verifying outputs, posing recall questions, and task prioritization.
2. Multi-Agent Decision Making: In line with newer systems like LLM-SmartAudit, this refers to the utilization of multiple AI agents (e.g., variants of GPT) to provide disparate perspectives of a given code.
3. Severity Scoring and Risk Ranking: Threats identified are placed in perspective based on severity, impacted functions, and execution paths utilized. A confidence score is generated to assist auditors with prioritization.

This reasoning process simulates the process of consideration performed by human security specialists, thus adding both precision and explainability.

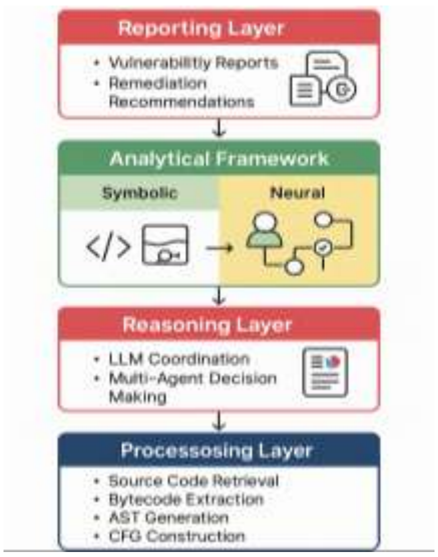
Reporting Layer

The last layer converts the technical output into human-interpretable interpretations and artifacts beneficial to developers and auditors:

1. Detailed Vulnerability Reports: Every vulnerability has a description in natural language, line references, impacted variables, and probable effect.
2. Severity Scores and Remediation Recommendations: From standard vulnerability taxonomies, i.e., the SWC Registry, the system calculates severity scores and recommends possible code changes.

This layer ensures that the framework is not only analytically robust but also practical for real-world adoption—bridging the gap between automated detection and developer action.

Fig. 2. Proposed Conceptual Framework for AI-Powered Smart Contract Auditing



Challenges And Open Issues

Despite the significant progress in applying AI and NLP techniques to smart contract analysis, several challenges remain unresolved. These challenges span across data availability, model generalization, explainability, and realworld integration.

Limited and Imbalanced Datasets

The development of strong AI models is dependent on the availability of large-scale, diverse, and annotated datasets. Unfortunately, the publicly available datasets on smart contracts have several limitations:

1. Small Size: Datasets such as ScrawlD and SmartBugs have at most a few thousand examples, which is too small to train contemporary deep learning models without overfitting.
2. Bias Towards Common Vulnerabilities: Vulnerabilities such as reentrancy and integer overflow are over-represented; on the other hand, access control, denial-of-service, or logical flaws are much under-represented.
3. Synthetic vs. Actual-World Data: Most datasets are either created synthetically or restricted to scholarly examples; therefore, they fail to reflect the complexity or coding patterns that occur in production DeFi contracts.
4. Inadequate Labeling Practices: Some datasets use binary labels for whole contracts, while others use vulnerability labels at the function or line level. This makes model transferability and benchmarking across studies challenging.

Generalization to Novel or Obfuscated Vulnerabilities

One of the main shortcomings of today's AI models is that they cannot handle zero-day exploits or hidden code. This is because:

- Pattern Overfitting: Supervised learning algorithms tend to memorize established patterns instead of acquiring general rules of vulnerability.
- Adversarial Robustness: Small syntax, variable name, or order of logic manipulations can deceive token-based or embedding-based models.
- Limited Semantic Understanding: Token embeddings and sequence models learn surface patterns instead of more abstract semantic understanding of contract logic and financial flow.
- This is calling for models that are able to reason about semantics and intent, and not simply syntax—maybe employing symbolic reasoning, logic programming, or hybrid models.

Integration with Real-World Development Pipelines

In spite of favorable research outcomes, there are limited AI-based audit tools incorporated into the development and auditing process of blockchain developers. The primary challenges are:

1. Lack of IDE Integration: Models are rarely deployed as plug-ins for tools like Remix, Visual Studio Code, or Hardhat.

2. Incompatibility with CI/CD Pipelines: Smart contract repositories often use continuous integration (CI) pipelines, but research models are not packaged for automation or dockerized deployment.
3. Security Auditing Lag: In practice, auditing is often performed manually after contract completion. Embedding AI tools into the write-compile-test-deploy lifecycle remains an open engineering challenge.

Evaluation Standards and Metrics

The testing of AI models to review smart contracts is non-standard and normally has substandard reporting quality:

1. Evaluation Misalignment: The majority of the evaluations fail to even test models on actual-world DeFi

contracts, failing to check robustness in actual blockchain environments.

2. Unreliable Experimental Settings: Different papers employ different training splits, datasets, and label types and therefore the comparisons are not fair.
3. Future research will use more rigorous and consensually determined criteria, possibly borrowing from the SWC Registry, Ethereum Bug Database, or specialized audit competitions.

Lack of Explainability

Explainability is an especially important, but under-researched, aspect of AI-powered auditing tools. Deep models are black boxes, and this poses numerous challenges:

1. Legal and Regulatory Transparency: Developers should justify their security decisions in business or legal terms. Flagging a vulnerability is not enough.
2. Human-AI Collaboration: Developers and auditors benefit from actionable information, such as identification of risky lines, impacted variables, or semantic triggers.

Computational Efficiency and Resource Constraints

AI models used in auditing often require high computational power, which limits real-time and scalable deployment. This section highlights the need for lightweight, efficient alternatives to enable broader adoption.

1. High Resource Intensity: Models like GPT or CodeBERT require high GPU/TPU resources for training as well as inference that may not be available for small teams.
2. Latency Concerns: Real-time smart contract editing or live audit situations are affected by inference time latency because of heavyweight models.
3. Scalability Problems: With increasingly complex smart contracts, processing time and memory requirements increase, slowing down and reducing audit efficiency.
4. Insufficiency of Lightweight Models: Optimized or lightweight models that are specifically designed for auditing smart contracts are sparingly available, constraining deployment in resource-constrained environments.

Legal, Ethical, and Regulatory Ambiguities

Deploying advanced models for contract auditing often demands considerable computing resources. This poses barriers to real-time use, especially in constrained or edge environment Such As:

1. Accountability Gaps: If there is an auditing error or an AI tool error, legal accountability is unclear—whether it should be borne by the developer, auditor, or tool owner.
2. Regulatory Compliance: The vast majority of AI models lack the transparency to accomplish financial or legal audit mandates for justification and traceability.
3. Data Privacy Issues: Accessing cloud-hosted AI tools for auditing risks compromising proprietary smart contract logic, creating intellectual property and confidentiality concerns.
4. Bias and Equity: Training sets that are skewed towards particular kinds of vulnerability can lead to uneven or misleading audit results from models.

Future Directions

To advance AI-driven smart contract analysis, future research needs to concentrate on developing robust, explainable, as well as flexible systems that can tackle a wide range of realworld issues. The following are necessary avenues for innovation:

Varied and Standardized Benchmark Datasets

A significant leap forward involves creating thorough, standardized datasets that represent the real-world environment of smart contract ecosystems. Datasets should contain a broad variety of vulnerability types, coding patterns, and obfuscation techniques, as well as manually verified by security experts annotations. By enabling community efforts, backed by academia and industry partners, one can encourage the development of open and reproducible benchmarking datasets enabling unbiased comparisons of AI-based tools.

Scaling Up Generalization using Transfer Learning

Future work should apply transfer learning and domain adaptation methods to enhance model performance on different smart contract platforms and programming languages. Fine-tuning models like CodeBERT and LLaMA pre-trained on blockchain codebases can possibly allow them to understand the semantics of Solidity or Vyper more effectively. In addition, meta-learning methods can facilitate quicker adaptation to new vulnerabilities.

Developing Explainable AI (XAI) Mechanisms

As AI takes on a more prominent role in security-critical tasks, explainability is critical. Explainable AI (XAI) techniques domain-specific, e.g., providing explanations for vulnerabilities, identifying risky code snippets, or offering counterfactual explanations, will need to be designed by researchers. Such transparency builds trust and allows for human-AI collaboration in auditing tasks.

Effective Models for Real-World Deployment

Scalability demands lean but resilient solutions. Future models will need to maximize computational efficiency via methods such as neural architecture pruning, knowledge distillation, or edge-friendly architectures. This would render them integratable within resource-constrained settings, for instance, IDEs, compilers, or blockchain nodes.

Integration with Development and Auditing Workflows

AI auditing tools should align with existing development ecosystems. Potential implementations include IDE plugins, CI/CD pipeline bots (e.g., GitHub Actions), and API-based vulnerability assessment services. Close collaboration with industry auditors will ensure these tools meet practical standards and usability requirements.

Blockchain Security through Federated Learning

Federated learning thus offers a promising path for collaborative model training of intelligent contract vulnerability models in decentralized environments without compromising sensitive code data centralization. Blockchain auditors or nodes can train local models and share contributions with a global model to enhance detection rates without compromising privacy and code confidentiality. Architectures and communication protocols appropriate for federated learning in distributed blockchain environments should be explored through future studies.

Cross-Platform Vulnerability Benchmarking

Considering the increasing heterogeneity of blockchain platforms such as Ethereum, Binance Smart Chain, and Solana, there exists an immediate necessity to develop cross-platform benchmarking tools and datasets. The future effort should focus on building universal metrics and platform-agnostic representations to evaluate the security posture of smart contracts in heterogeneous settings. These advancements will allow for uniform evaluation and support models more generalizable.

DISCUSSION

The integration of Artificial Intelligence in auditing smart contracts is a paradigm shift towards greater security and autonomy in the blockchain domain. The results of the research survey show a definite shift—away from traditional rule-based systems towards progressively smarter and more flexible models driven by deep learning and transformer architecture.

Among the prevailing themes that emerge from the literature is the trade-off between performance and interpretability. While deep neural networks and large language models such as CodeBERT, LLaMA, and GPT models show impressive performance gains in vulnerability detection, their lack of transparency restricts explainability. For applications such as smart contract security where legal traceability and auditability are critical, a lack of explainability can preclude real-world usage.

Another significant trend that has emerged is the unification of hybrid and multi-modal systems, which combine the traditional symbolic features, like Abstract Syntax Graphs (ASTs) and Control Flow Graphs (CFGs), with neural features. These systems normally perform more effectively with different types of vulnerabilities and obfuscation methods; with increased computational expense and resource usage, however, hybrids prove to be a problem when used in real-time applications or in light-weight development environments.

The same constraints are applicable to datasets as well, the majority of models have depended on artificially created datasets, or narrowly founded datasets that fail to capture the details and the complexity of actual world smart contracts. This translates to other constraints in regards to common sense parameters without capturing meaningful overview generalization of models and raising overfitting issues. Limited common based assessments and viable benchmarks that can measure different measures of evaluation in other research creates this even larger challenge. The recent emphasis on explainability and embedding workflows in research that followed indicates a natural tendency towards proving potential for plausible adoption of AI based approaches in the real world, there's still implementation gap in evidence, for example, many proofs of concept so far but no integrations into IDE or IDE runs in continuous integration pipelines as auditing tool chains. The challenge will be in transitioning from proof of concept, to value add or pay off for actually utilized AI based solutions in regard to smart contract security, and realizing the full break-through of AI potential beyond theoretical break-throughs.

Another area drawing increasing attention is the use of multi-agent LLM systems for collaborative auditing. By distributing the audit process across several specialized AI agents—each trained for a different vulnerability type or contract pattern—researchers aim to simulate the layered decision-making process of human auditors. This not only boosts detection robustness but also introduces redundancy for error correction. However, coordination between agents, communication overhead, and conflict resolution strategies remain open challenges. Moreover, the concept of adaptive learning, where auditing models continuously improve through user feedback or evolving code trends, is still in its infancy. While such systems provide immense potential for the development of self-refreshing security infrastructures, they also bring in substantial and serious threats, including threats of data poisoning, audit drift, and malicious input tampering. Aside from these issues, it is also fascinating to observe that much of the software that has been discussed and referenced in current literature is open-source or non-reproducible. This inherent quality significantly makes it much harder for the wider research community to extend, validate, or perform experimental comparisons of these software solutions.

Moreover, when we are heading towards an age where smart contract audit tools are in the process of being approved by regulations, it becomes essential for developers to actively deal with a variety of socio-technical issues. These issues include serious concerns like user trust, legal liability of false positives and false negatives, and presence of training bias, which can inadvertently lead to harm to some contract authors. While such socio-technical issues are normally under-emphasized, they are actually critical to the successful development of auditing ecosystems that are not only trustworthy but even inclusive and legal compliant.

CONCLUSION

The increasing reliance on smart contracts in different blockchain implementations highlights the importance of scalable, smart, and secure audit mechanisms. While rulebased analytical systems are beneficial, they cannot recognize sophisticated or yet unrecognized vulnerabilities. Therefore, the application of Artificial Intelligence—Natural Language Processing and deep learning, in particular—has emerged as a promising path towards improving the dependability and security of smart contracts.

This paper has provided a comprehensive overview of artificial intelligence-based methods for analyzing smart contracts. We have categorized the dominant methods into four broad categories: AI-supported rule-based methods, supervised code embedding-based models, transformer language models, and hybrid multi-modal frameworks. A comparative analysis was provided to describe the intrinsic strengths and weaknesses of each method. We have also elaborated on a conceptual layer model, with the vision of integrating the analytical benefit of these diverse methods.

While significant progress has been made, there remain several major challenges to the discipline. These challenges include limited access to high-quality datasets, models prone to generalization failures, limited interpretability, challenges in collaboration with present development toolchains, and inefficiencies in computations. Breaking these barriers is crucial for successfully transferring theoretical work to practical use. In the future, some of the areas that we have considered for future research are building standardized benchmarks, transfer learning, explainable AI innovation, and the deployment of light models in edge settings. If these areas are targeted, future research can facilitate more robust, efficient, and adaptive smart contract analysis tools, ultimately making the blockchain ecosystem more resilient and secure. Apart from that, with the advancing blockchain environment, the need for proactive and automated security will continue to increase. The move towards decentralized finance (DeFi), NFTs, and business blockchain applications is driving smart contract sophistication that's out of the reach of traditional tools. AI-powered solutions bring with them not only scalability but the ability to keep pace with new attack vectors, making them a necessity for next-gen auditing.

REFERENCES

1. L. Brent, A. Jurisevic, and B. Scholz, “Vandal: A Scalable Security Analysis Framework for Smart Contracts,” arXiv preprint arXiv:1809.03981, 2018.
2. J. Tikhomirov, E. Voskresenskaya, and E. Marchenko, “SmartCheck: Static Analysis of Ethereum Smart Contracts,” in Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain, 2018, pp. 9–16.
3. Y. Liu et al., “Smart Contract Vulnerability Detection: A Deep Learning Based Method,” in IEEE Access, vol. 8, pp. 219398–219407, 2020.
4. A. Tereshchenko and E. Komleva, “Detecting Vulnerabilities in Smart Contracts Using CodeBERT and NLP Techniques,” in 2022 IEEE International Conference on Blockchain (Blockchain), pp. 1–8.
5. N. Khodadadi and H. Tahmoresnezhad, “A Multi-modal Deep Learning Framework for Smart Contract Vulnerability Detection,” in Computer Standards & Interfaces, vol. 85, 2023.
6. Y. Mi et al., “Byte-level Vulnerability Detection with Metric Learning for Ethereum Smart Contracts,” in IEEE Transactions on Software Engineering, Early Access, 2024.
7. Z. Yu et al., “A Retrieval-Augmented Generation Model for Explainable Smart Contract Auditing,” arXiv preprint arXiv:2401.02345, 2024.
8. Q. Wei et al., “LLM-SmartAudit: Multi-Agent GPT for Smart Contract Auditing,” in NeurIPS Workshops, 2024.
9. T. De Baets, R. Vande Ginste, and F. De Backere, “A Survey of Machine Learning-Based Smart Contract Security Analysis,” in Journal of Systems and Software, vol. 194, 2023.
10. M. Ozdag, "On the Challenges of Smart Contract Vulnerability Detection Using NLP and Machine Learning," Journal of Blockchain Research, vol. 5, no. 2, pp. 45–61, 2025.
11. Z. Jin et al., “SmartEmbed: A Context-Aware Embedding Model for Smart Contract Code Understanding,” in IEEE Transactions on Software Engineering, 2022.
12. D. Grechishnikov et al., “AI-based Code Completion and Error Detection for Solidity Using Transformers,” in Proceedings of the 2022 International Joint Conference on Neural Networks (IJCNN), pp. 1–7
13. A. Nguyen et al., “Graph Neural Networks for Smart Contract Vulnerability Detection,” in Proceedings of the 30th ACM Conference on Computer and Communications Security (CCS), 2023.
14. G. Chen et al., “Eth2Vec: A Token Embedding Approach for Smart Contract Semantics,” in Information and Software Technology, vol. 145, 2022.
15. S. Qiu, H. Liu, and J. Zhang, “Challenges and Directions in Smart Contract Testing: A Survey,” in IEEE Access, vol. 10, pp. 120413–120429, 2022.
16. L. Zhang and K. Li, “Federated Learning for Privacy-Preserving Smart Contract Analysis,” in IEEE Internet of Things Journal, Early Access, 2024.
17. M. Elsabagh et al., “Cross-Platform Smart Contract Analysis: Benchmarks and Pitfalls,” in Proceedings of the 2023 IEEE International Conference on Blockchain, pp. 94–102.

18. H. He et al., “Transfer Learning for Solidity Smart Contracts with Code Summarization,” in Proceedings of the 2022 IEEE Symposium on Security and Privacy, pp. 1142–1155.
19. A. Kumar and S. Roy, “Explainable AI for Blockchain Smart Contract Auditing: Opportunities and Gaps,” in Proceedings of the 2023 International Conference on Trust, Privacy, and Security in Digital Business, pp. 80–95.
20. B. Tan and Y. Fang, “Deploying AI Auditors into CI/CD Pipelines: A Smart Contract Security Perspective,” in ACM Transactions on Software Engineering and Methodology, vol. 32, no. 1, 2024.