High-Confidence Computing 6 (2026) 100374

Contents lists available at ScienceDirect

High-Confidence Computing

journal homepage: www.sciencedirect.com/journal/high-confidence-computing

Research article

Explainable DNN for smart contract vulnerability detection in the Metaverse

Ebuka Chinaechetam Nkoro [a], Love Allen Chijioke Ahakonye [b], Dong-Seong Kim [c,d,*]

[a] Computer and Information Sciences Department, Towson University, MD 21218, Towson, USA
[b] ICT Convergence Research Center, Kumoh National Institute of Technology, Gumi 39177, South Korea
[c] IT Convergence Engineering, Kumoh National Institute of Technology, Gumi 39177, South Korea
[d] NSLab Co. Ltd., Kumoh National Institute of Technology, Gumi 39177, South Korea

ARTICLE INFO

Article history:
Received 23 May 2025
Revised 15 October 2025
Accepted 3 November 2025
Available online 11 November 2025

Keywords:
Blockchain
Metaverse
Cybersecurity
Deep learning
Explainable AI
Smart contract
Machine learning
Vulnerability detection
Smart contract analysis
Ethereum smart contract
Blockchain vulnerabilities
LLMs

ABSTRACT

Smart Contracts (SCs), which are the backbone of automated transactions and digital assets within the Metaverse, ironically suffer from their own share of security vulnerabilities. While detecting these SC vulnerabilities using Artificial Intelligence (AI) and Deep Neural Networks (DNNs) has demonstrated remarkable performance and gained wide adoption, a critical limitation remains: the lack of explainability in these black box models. To facilitate meaningful progress in this field, our study addresses this gap by introducing a model-agnostic explanation framework that is both visual and quantitative, with human stakeholders actively involved to govern, verify, and interpret SC model predictions. The explainable SC outputs can be utilized for reward issuance and digital assets governance in the Metaverse. The effectiveness of our proposed Explainable AI (XAI) approach is validated using benchmark datasets, BCCC SCsVul 2024 and BCCC SCsVul 2023, comprising Ethereum SC entropy source codes, where it achieves an optimal detection accuracy of 97.13% alongside comprehensive explainability. To the best of our knowledge, this represents the first attempt at making Ethereum SC vulnerability detection within the Metaverse explainable, offering a valuable foundation for blockchain researchers, Metaverse security experts, and practitioners seeking verifiable, trustworthy, and auditable Ethereum SC vulnerability detection.

1. Introduction

The Metaverse [1], the future of an immersive and rewarding Internet, has evolved in design, innovation, and cybersecurity challenges. The Metaverse, by definition, bridges the physical world and virtual world to allow users to enjoy an immersive environment [2,3]. One of the inherent challenges of the Metaverse is its digital economy sector, built upon Web3 and cryptocurrency. Within the Metaverse, immersive actors and other stakeholders can participate and obtain Non-Fungible Tokens (NFTs) as rewards for their contributions [4].

To fundamentally balance the speed and scale of secure transactions within the Metaverse [5], Smart Contracts (SCs) [6]. SCs are integrated into blockchain technologies to self-execute upon completion of a configured set of conditions. Once an SC is executed, the process is irreversibly recorded in the blockchain [7].

However, SCs, although "smart" [8], are not immune to security risks, as successful attacks on vulnerable SCs can result in severe financial losses in the Metaverse. For example, an attacker can exploit a flaw in a marketplace SC and transfer expensive NFTs to themselves without paying [9]. Recent drainer operations often blend traditional tactics, such as phishing and social engineering, with malicious executable SCs for vulnerable users and groups. A recent example is attributed to the stealthiest crypto hack and heist by North Korea's notorious Lazarus Group, in which 1.5 billion USD in Ethereum tokens were stolen from Bybit's wallet [10]. The hackers exploited and executed vulnerable SC codes, which allowed the attackers to bypass multi-signature security measures and drain Bybit's wallet undetected. At a global scale, an estimated amount of 2 billion USD related to illicit cryptocurrency was accounted for by leading security auditors such as Chainalysis, reflecting the need for enhanced defence, real-time detection, and resilience against vulnerable SCs.[1]

Besides crypto theft, the motive behind executing vulnerable SCs for financial gain spans broader areas such as funding, facilitating, and targeting vulnerable groups with minimal traceability [11]. Thus, security experts have raised several alarms

* Corresponding author.
E-mail address: dskim@kumoh.ac.kr (D.-S. Kim).

[1] https://www.chainalysis.com/blog/bybit-exchange-hack-february-2025-crypto-security-dprk/

https://doi.org/10.1016/j.hcc.2025.100374

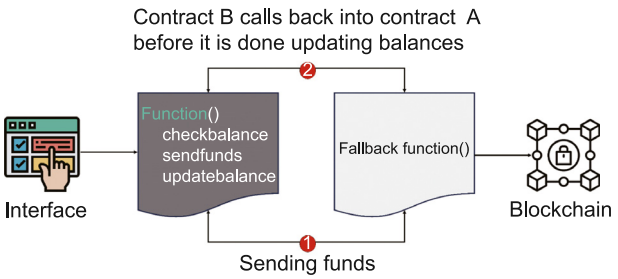

Fig. 1. Illustration of a sample reentrancy attack in a Solidity SC vulnerability, which exploits improper ordering of operations, allowing an attacker's contract to repeatedly probe a vulnerable function before balance updates, potentially draining all funds.

about its categorized vulnerabilities [5,7,8], such as execution & logic flaws, and machine-specific weaknesses, which in general aim towards compromising its confidentiality, integrity, and availability (CIA triad) [12], as exemplified in Fig. 1.

Meanwhile, conventional (static and symbolic) SC vulnerability detection tools (like Slither, Smart Check, Manticore, and Mythril) employed for capturing vulnerability patterns in SC codes suffer from various limitations, such as human limitations, increased false positive rates, path explosion [13], and limited coverage in the presence of extensive data [14]. Due to the limitations of conventional SC vulnerability detection, recent methods have employed Artificial Intelligence (AI) and Machine Learning (ML) to aid in intelligence-driven detection of SC vulnerabilities and anomalies [15–17].

Despite the utility of Deep Learning (DL) methods in SC vulnerability detection, various bottlenecks still hinder trustworthy detection, monitoring, and interrogation of vulnerable SC codes. While previous works have focused mainly on improving detection accuracy, a key threat to DL models is the lack of model interpretability methods that can provide visually interpretable predictions about model predictions, bolstering confidence, detection speed, and trust for security experts while improving the overall security of SC detection [18,19]. Although existing literature has begun studying XAI for cybersecurity, there remains a lack of research addressing the reliability and explainability concerns in AI-enabled SC vulnerability detection to enhance security expert decision-making, auditing, and model trustworthiness [12].

1.1. Research contributions

For brevity, the contributions of this study are highlighted as follows:

- This article studies the vulnerability of SC vulnerability detection systems within the Metaverse, and reveals the unique yet pressing challenge of model trustworthiness and interoperability. The proposed study aims to make predictions of DL-enabled SC vulnerability detection systems auditable and verifiable, which can improve overall model robustness, interpretability, and compliance with the growing global demands for trustworthy AI model utility.
- In this study, XAI seeks to answer the simple but important question: **"Why should I trust the model predictions of a SC vulnerability detection system in the Metaverse?"**. Therefore, to make DL-enabled SC detection more interpretable and trustworthy, an explainable AI (XAI) method using model-agnostic explainers, such as LIME and SHAP, serves as an additional layer of trustworthiness and reliability for verifiable predictions of vulnerable SCs from benign code patterns. We equally propose a high-confidence SC vulnerability detection scheme, which entails estimating and calibrating predictive uncertainty when distinguishing vulnerable from benign contracts through benchmark, model-agnostic XAI methods. The proposed framework enables security teams to enforce confidence thresholds that support risk-aware defense and mitigation against vulnerable SCs.
- Compared to previous studies, and to our knowledge, this study represents pioneering literature, demonstrating, with benchmark datasets (entropy and raw code features) and results, how visual and quantitative XAI methods optimally enhance interpretable SC vulnerability detection systems within the Metaverse.

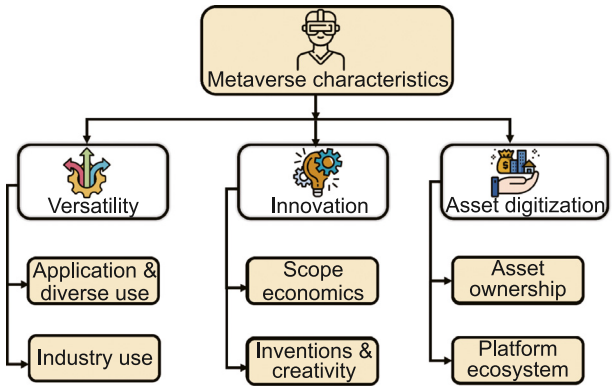

Fig. 2. Illustration of the metaverse characteristics.

1.2. Paper organization

The paper is organized as follows: Section 1 is the introduction, and Section 2 consists of related works within SC vulnerability detection and the need for XAI integration. Section 3 provides an overview of the proposed method, with description, feature extraction, and processing works on the dataset and XAI model. Section 4 presents the model and XAI experimental results, while the conclusion is presented in Section 5.

2. Background & related works

2.1. Digital economy dynamics in the metaverse

The Metaverse, as inspired by Neal Stephenson [20], is intended to be the new phase of an immersive Internet in which physical users or objects, such as avatars, interact boundlessly with virtual worlds [1]. Recent authors have categorized the core characteristics of the Metaverse according to its *application, diverse use, innovation, and asset digitization* [2] as shown in Fig. 2.

Each category harmonizes in functionality to constitute the goal of a rewarding, fun, and immersive Metaverse. The Metaverse archives its fully immersive environment with diverse ICT technologies such as IoT sensors, wearable technologies, 5/6G networks, AI, and blockchain technology [3].

Beyond gaming, value creation in the Metaverse has gained wide attention. Specifically, users, businesses, and participants can engage in virtual products and services to capture stronger interactions and connections, market audiences, and attract more customers [21]. They also gain NFTs as rewards with play-to-earn games and Metaverse marketplaces for trading assets or virtual properties (e.g. land parcels, wine, holiday trips, and arts) [22].

The NFTs issued in the Metaverse serve not only as rewards but also as personalized identity tokens for Metaverse users. Thus, confidentiality, integrity, and availability of digital assets within the Metaverse have become essential for its security, trustworthiness, and quality of experience [23]. Therefore, recent authors have emphasized the need for digital asset security within the Metaverse with SCs. [5,24]. The following subsection deep dives into the nature of SCs, and its vulnerability detection methods.

2.2. SC vulnerability detection

SCs are termed 'smart' because predefined codes without the involvement of any intermediary can automate the execution of a digital contract with speed, efficiency, and accuracy [25]. However, detecting vulnerable SC codes from benign patterns has become an essential challenge in recent years [26,27]. Based on existing methods, SC vulnerability detection can generally be grouped into conventional and AI methods [18].

Conventional SC vulnerability detection tools, such as Oyente, Osiris, Mythril, DEPOSafe, ZEUS, and Securify, generally analyze SCs at the bytecode level. A significant drawback of conventional SC detection methods, as reported in previous literature, is the computational difficulty and increased false positives in examining all possible SC code paths, which can potentially allow vulnerabilities to go undetected. For example, a recent conventional SC vulnerability detection framework employed $\mu-$ Calculus rules for SC vulnerability detection [28]. Although their proposed method showed excellent results with a limited 40 vulnerable and clean SC dataset, the detection strength with an increased SC code volume is not discussed. Another recent conventional method conducted a quantitative analysis and vulnerability scanning of SC codes with 18 conventional vulnerability scanners [29]. A significant limitation identified in their work is the reliance on static scanning capabilities within specific codebases and a strict dependency on specific compiler versions. The highlighted static SC detection issue is evident in the broad range of conventional tools employed, such as SmartCheck, Slither, Oyente, Maian, Artemis, Osiris, Mythril, Securify2, TeEther, ConFuzzius, sFuzz, Smartian, GNNSCVulDetector, and MANDO-GURU. Other similar works [30–32] have similarly employed static tools for SC vulnerability analysis and detection, yet the demand for scalable and more intelligent detection is evident.

Given the growing complexity of SC codes and vulnerability detection mechanisms, security teams can benefit from AI methods for automated detection and SC security. Compared to conventional tools, DL frameworks can extract semantic SC code features and correlations to detect vulnerabilities in SCs. Current studies in [33] proposed Vulnsense, a multimodal SC feature framework SC vulnerability detection using DL Graph Neural Networks (GNNs), bidirectional long-short-term memory (LSTM), and bidirectional transformer encoder representations for improved detection accuracy. Their proposed method achieves an average accuracy of 77% within benchmark datasets like Smartbugs Curated, SolidiFI-Benchmark, and Smartbugs Wild. Another study using GNNs and expert patterns under subspace mapping employed bytecode and opcode features for SC reentrancy vulnerability detection [34]. A similar study employed a SC vulnerability detection scheme using GNN and a private dataset [35]. Their proposed framework detected three types of SC vulnerabilities: reentrancy, timestamp, and infinite loop attacks while achieving an accuracy of 60.53%, 60.81%, and 71.89% with conventional detection tools such as Oyente, Mythril, and Securify. On the other hand, a better performance of 91.14%, and 92.92%, respectively, was achieved when using GCNN and GNNs. Authors in [36] employed the *Smartbugs Wild* dataset for SC vulnerability detection, using GNNs to perform SC vulnerability detection at the bytecode level only and attained an accuracy of 95.49%.

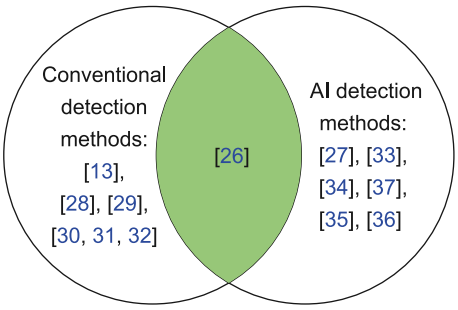

Fig. 3. Summary of discussed conventional and AI-based SC vulnerability detection methods.

To capture more semantic features of SC codes, recent works have employed three DL models, namely, CodeBERT, optimized LSTM, and optimized CNN, for improved model performance [27]. Given the study's focus on accuracy improvement, the CodeBERT model attained the highest accuracy and detection rate for SC vulnerability detection. Other works have proposed a multimodal fusion of SC source codes to capture diverse SC code patterns for effective detection and improve detection performance. They specifically utilized a Text-CNN with an attention module for a feature aggregation network with built-in spatial attention units to extract semantic features of the SC codes for detection [37].

Together, the conventional and AI methods for SC vulnerability detection, as presented in Fig. 3, have primarily focused on accuracy improvement. However, they often neglect the lack of interpretability and foster blind trust in black-box models, which can output false predictions. This severely undermines detection efforts—a challenge we aim to tackle. Regarding model interpretability issues, modern systems relying on AI are now required to comply with explainability methods to foster stakeholder transparency and reliability [38]. XAI, as proposed in this work and the jurisdiction of SC vulnerability detection, fosters trust and aids security personnel in interpreting, refining, training, and improving the decision-making process of AI-enabled SC vulnerability detection, and not blindly trusting them.

2.3. The need for XAI in SC vulnerability detection

Despite recent advances in SC vulnerability detection, the integration of XAI remains relatively underexplored [18]. AI-driven models and traditional analysis tools often struggle with interpretability, offering little insight into *why* a specific SC prediction was made [33–35,37]. Recent efforts have introduced explainable techniques capable of providing meaningful justifications for general domain ML model predictions [39–41]. XAI taxonomy is presented in Fig. 4, where we categorize its utility and functionality according to **time, nature, and scope** of explanations. Most studies have adopted post-hoc explanations like ours due to the computational requirements of explaining models during training. For instance, recent studies in the domain of malware analysis have utilized feature pattern datasets, including memory management and CPU usage, alongside traditional classifiers to detect and explain malware patterns [42]. By leveraging XAI libraries such as Local Interpretable Model-agnostic Explanation (LIME) and SHapley Additive exPlanations (SHAP), their proposed approach successfully explained malware behavior through visualizations indicating low, high, and abnormal patterns of suspicious CPU activity. While traditional ML classifiers are inherently interpretable and demonstrate strong accuracy on limited datasets, their work does not address model explainability within complex black-box ML algorithms, which are not easily

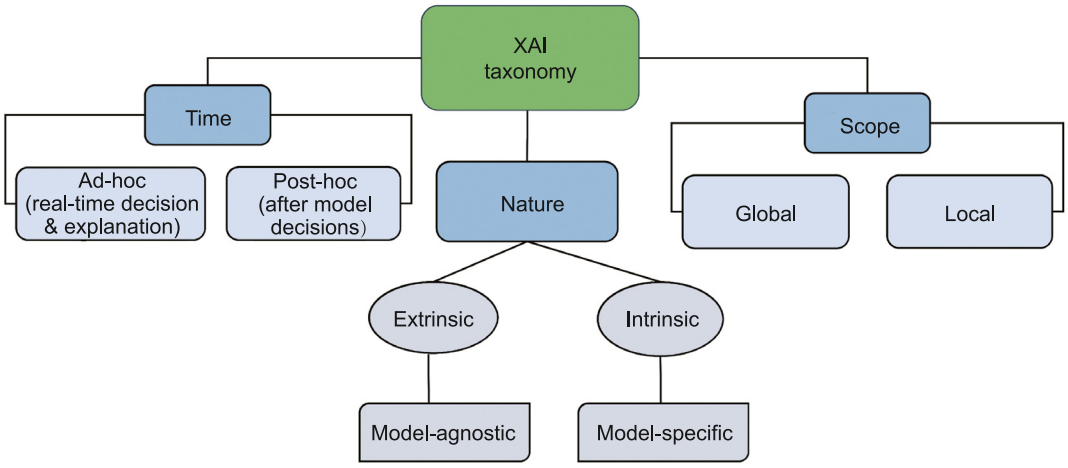

Fig. 4. A taxonomy of XAI methods based on time, nature, and scope, outlining when, how, and where explanations are applied.

interpretable and are essential for capturing semantic feature representations in the context of large-scale datasets. Similarly, XAI libraries like SHAP and LIME have been employed by Common Weakness Enumeration (CWE) experts to help clarify why a specific CWE label was assigned, especially when evaluating code security in critical cyber–physical systems. The XAI code representation provided in the study demands substantial technical knowledge and domain expertise for proper interpretation, in contrast to more easily readable and visually interpretable approaches that cater to non-experts.

Another study utilized a GCNN classifier and graph-based XAI visualization to detect the vulnerabilities in assembly code [43]. A key drawback of their approach is that it is hinged on an arcane and high technical comprehension of the visualization subgraphs and nodes, which is not easily readable for XAI stakeholders. Closely related literature has primarily focused on highlighting feature importance rankings using the LIME explainer for fraud detection in blockchain transactions [44].

To the best of our knowledge, no prior work has effectively integrated XAI techniques into SC vulnerability detection. While related studies have explored XAI for general blockchain fraud detection [44] primarily through feature importance methods such as LIME, they fail to convey model certainty or offer intuitive, human-centered visual explanations. In contrast, our work is the first to bridge this gap by introducing a dedicated XAI framework explicitly tailored for SC vulnerability detection, combining visual and quantitative interpretability to support deeper understanding and trust for blockchain SC stakeholders.

3. Proposed system and research methodology

This section describes the methodology for detecting SC vulnerabilities with XAI. The high point of this research is the explainability, transparency, and intelligibility of SC vulnerability detection models in the Metaverse.

Attacks targeting real-time transaction execution in virtual events pose unique risks compared to traditional DeFi attacks [3, 45]. DeFi platforms focus on financial vulnerabilities but virtual events depend on seamless, low-latency transactions to ensure user engagement and smooth interaction [3]. Disruptions in virtual transactions can severely affect user experience, leading to immediate loss of trust, engagement, and potential economic damage due to decreased revenue and reputational harm [2]. These attacks also exploit scalability and latency issues more acutely, destabilizing systems reliant on high-throughput interactions, such as ticketing or auctions [2]. Additionally, the presence of digital assets like NFTs and the time-sensitive nature of virtual events further exacerbate the impact of such disruptions [46]. Ultimately, attacks on real-time transactions in virtual events go beyond financial loss, threatening platform stability, user confidence, and long-term ecosystem growth.

3.1. Metaverse-specific threats

Unlike conventional blockchain-based SCs, where the threat model primarily focuses on financial loss from code exploitation, threats to SCs within the Metaverse expand dramatically. Primarily, these SCs are designed for dynamic governance in the virtual world, managing cryptocurrency transactions, handling unique digital assets such as NFTs, avatars, and virtual properties [45]. For example, an SC for managing a virtual asset or Soulbound Tokens (non-transferable), e.g., an "NFT sword or unique skin", can be manipulated by an attacker to orchestrate a reentrancy attack where duplicates of these assets are created by recursively invoking the withdrawal function before the contract's state is updated. Another example can be related to 'real-time transaction execution' for virtual events in the Metaverse, where an attacker can fill a block with high-gas transactions to prevent a time-sensitive arbitrage or liquidation transaction from being processed. The impact of a missed financial opportunity or a loss for a specific trader can result in disruptions that degrade the user experience and damage platform reputation. This is unlike traditional DeFi attacks, where the consequences are often limited to only financial loss.

3.1.1. Differentiation from DeFi

In the Metaverse, SCs go beyond financial transactions, playing a key role in virtual social interactions, digital asset governance, and identity management. These functions introduce unique security challenges not found in traditional DeFi environments [47]. Metaverse SCs govern assets such as NFTs, virtual property, and avatars, facilitating activities including minting, asset transfers, voting during events, and ticket validation [48]. These require real-time execution across multiple decentralized platforms, adding complexity to security. Unlike DeFi SCs, where vulnerabilities often involve financial theft, Metaverse SCs are vulnerable to attacks targeting user identities, disrupting virtual events, or enabling unauthorized access to digital assets. Additionally, issues such as cross-chain interoperability and privacy concerns, including the exposure of avatar identities, further complicate security. Given these factors, the security needs of Metaverse SCs differ significantly from DeFi SCs, requiring a tailored approach to vulnerability detection that accounts for virtual interactions, real-time transactions, and identity management.

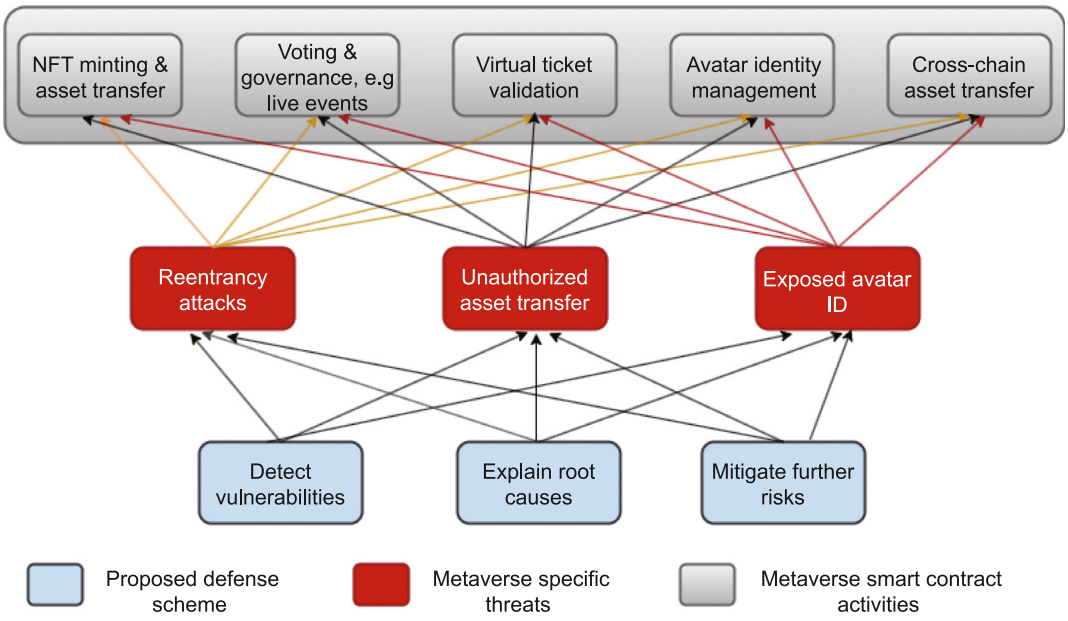

Fig. 5. Illustration of attack surfaces of SC vulnerability detection within the Metaverse, and proposed defence strategies.

3.2. Problem definition

Before diving into the proposed technique, we present an outlined justification of threats and challenges accompanying Metaverse SCs and the relevance of the proposed defense scheme as shown in Fig. 5. Typical Metaverse SCs are characterized by minting and transferring virtual assets, voting during live transactions, and virtual ticket validations. Unlike generic blockchain SCs, Metaverse SCs are specifically designed to address unique challenges in the Metaverse, such as identity management of avatars [49,50], real-time transaction execution for virtual events [51], interoperability across multiple blockchain platforms for seamless asset transfers, and enhanced privacy protections for user identities in virtual spaces. Although Metaverse SCs seamlessly manage digital assets within virtual environments, they also introduce distinct threats, including unauthorized asset transfers due to vulnerabilities in NFT minting contracts, reentrancy attacks targeting real-time marketplace transactions, and privacy breaches exposing avatar identities, leading to asset loss across platforms. Thus, there is a critical need to ensure these SCs are secure, with vulnerabilities effectively detectable and explainable through our proposed defense scheme to mitigate these Metaverse-specific risks.

3.2.1. Theoretical motivation

Given the complexity of SC code within the Metaverse, the task is to efficiently detect and explain vulnerabilities that could lead to unauthorized transfers of virtual assets or compromise digital identities. Let $D = \{d_i\}_{i=1}^{N}$ be a dataset of SCs, where each contract d_i is represented by a feature vector $x_i \in \mathbb{R}^d$. The objective is to classify each contract as vulnerable or non-vulnerable based on features extracted from the contract (e.g., bytecode entropy). Formally, the classification loss is given by Eq. (1).

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(f(x_i, \theta) \neq y_i), \tag{1}$$

where $\mathbb{I}(\cdot)$ is the indicator function, $f(x_i, \theta)$ is the model prediction, and y_i is the true label.

Beyond classification, we formalize $f(x_i, \theta)$ explanations as functions that assign importance scores to input features, as inspired by authors in [52]. Let the classifier be $S : \mathbb{R}^d \to \mathbb{R}^C$, where C is the number of classes (vulnerable vs. non-vulnerable), and let the explanation method be $E(S, x) \in \mathbb{R}^d$, assigning to each input x a feature attribution vector representing the contribution of each feature to the model's prediction. This framework provides an adaptable foundation for designing an accurate, explainable SC model for SC vulnerability detection in the Metaverse.

Furthermore, detecting vulnerabilities in Metaverse SCs is more challenging than in traditional blockchain systems due to their dynamic and high-dimensional nature. For example, a single SC can be designed only for governance against banned words [48]. Unlike static DeFi contracts, Metaverse SCs frequently evolve. They integrate new virtual assets and interactions across decentralized platforms [48,53]. This necessitates continuous updates to detection models to identify emerging vulnerabilities and attack vectors, especially those involving avatars and virtual assets. The high dimensionality of Metaverse SCs [54], encompassing bytecode, asset management, and user interactions, further complicates the detection of vulnerabilities. Effective real-time processing, classification, and *explanation* of this complex data is critical, particularly during high-stakes events like live Metaverse auctions or ticket sales. Ultimately, the rapidly changing nature of Metaverse SCs requires adaptive, explainable, and real-time detection methods to manage their complexity and ensure secure virtual interactions.

3.3. Proposed technique

The proposed technique in Fig. 6 presents and addresses Metaverse-specific SC vulnerability detection, which is designed with accountability and governance wherein human stakeholders (SC developers, blockchain experts, affected users, other AI users, regulatory teams) with requisite expertise are kept in the loop to govern, remediate, flag, and inspect SC model decisions.

The first stage begins with proper processing and feature extraction of SC vulnerabilities from source codes (compiler-based and non-compiler-based categories). This extraction process is achieved with the SCsVulLyzer V2.0 into a statistical CSV format [55]. The next stage involves further analysis of the preprocessed data by domain experts in AI and blockchain, training of the DL model, and an explanation of the predicted SC outputs

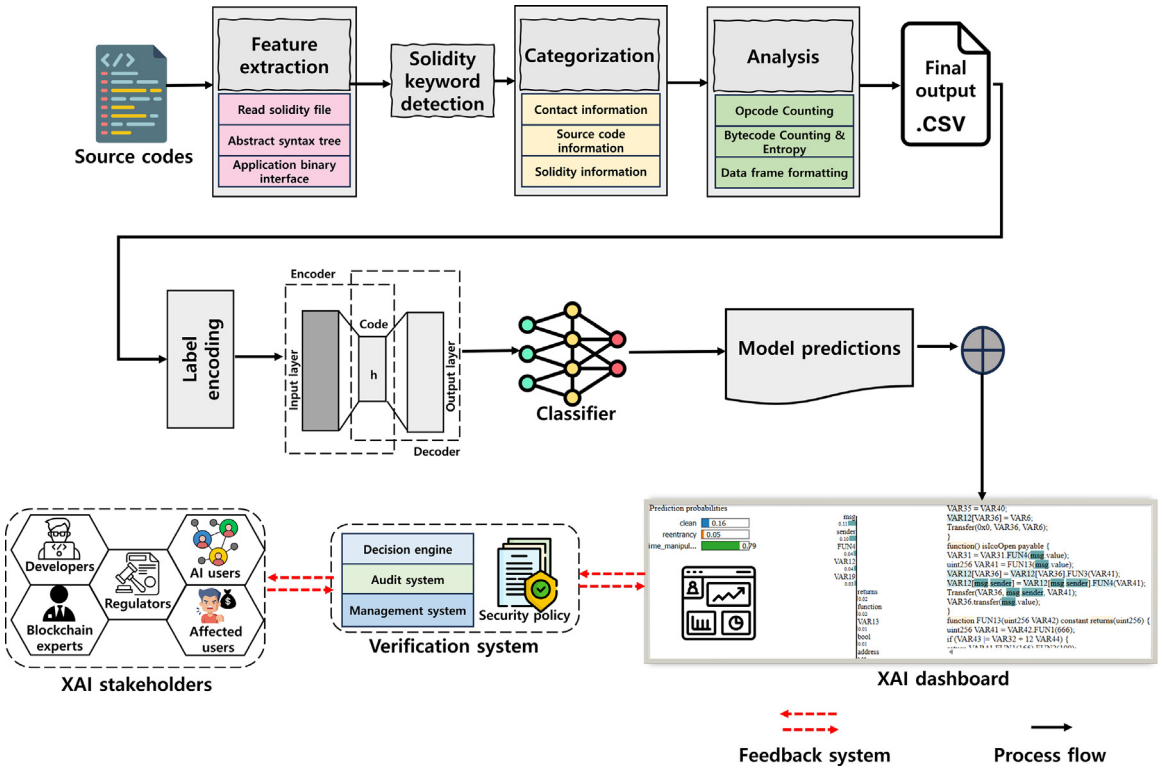

Fig. 6. Illustration of the proposed XAI SC vulnerability detection system.

using a post-hoc XAI tool. Integrating XAI complements stakeholders' expertise, enabling them to understand and verify model predictions. This method reduces both false positives and false negatives, ensuring that true vulnerabilities are more reliably detected before SCs are deployed. This is crucial in immutable blockchain environments.

This section describes the methodology for detecting SC vulnerabilities with XAI. The high point of this research is the explainability, transparency, and intelligibility of SC vulnerability detection models in the Metaverse.

Authors in [55] utilized the SCsVulLyzer V2.0 to analyze Ethereum SCs, classifying features into compiler-based and non-compiler-based categories. It introduces contract, source code, and Solidity information to quantify function counts, loops, and lines of code for deeper analysis. Bytecode entropy in Eq. (2) measures the randomness of bytecode, enabling the assessment of unpredictability and complexity, particularly in cryptography and anomaly detection.

$$H(X) = -\sum_{i=1}^{n} p(x_i)\log_2 p(x_i), \tag{2}$$

where X is the set of bytes, $p(x_i)$ is the probability of each byte occurring, calculated as the frequency of x_i divided by the total number of bytes. For a byte array d of length N, the entropy $H(d)$ is calculated in Eq. (3).

$$H(d) = -\sum_{x \in d} \left(\frac{\text{count}(x)}{N}\right) \log_2 \left(\frac{\text{count}(x)}{N}\right). \tag{3}$$

The label of the resultant data is encoded for model predictions. The model decision is used for explainability and is accessible on the XAI dashboard for decision-making.

3.4. Auto encoder (AE) for source code dimensionality reduction

Given the dense spectrum of compiler-based and non-compiler-based SC features spanning syntax trees, ABI definitions, code structure metrics, and natural language features, SC representations become inherently high-dimensional. This high dimensionality poses challenges during model training, such as increased computational complexity and high dimensionality. To mitigate this challenge, the AE is employed as a pre-training step. Specifically, the AE, consisting of the encoder and decoder, compresses the SC high-dimensional input into a lower-dimensional latent space, which enhances the learning of latent patterns and correlations between features and maintains performance.

The AE encoder function f_{enc} and the decoder function f_{dec} are defined in Eq. (4):

$$\text{Encoder: } z = f_{\text{enc}}(x), \quad \text{Decoder: } x' = f_{\text{dec}}(z), \tag{4}$$

where x represents the input data, z is the compressed representation in the latent space, and x' is the reconstructed data. The loss function used in training the AE encourages the model to learn a representation of the data that retains as much helpful information as possible. The reconstruction error, as shown in Eq. (5), is minimized during training:

$$\mathcal{L}\text{AE} = \frac{1}{N}\sum i = 1^{N} |x_i - x_i'|^2. \tag{5}$$

In this configuration, the AE is trained using the mean squared error loss function, optimized with the Adam optimizer at a learning rate of 0.001 for a maximum of 50 epochs and a batch size of 64, with early stopping enabled to avoid overfitting. The AE is adopted as a superior choice in this context, compared to traditional feature selection methods [55], which may permanently discard source code entropy features deemed less important. The AE retains all input features by learning compressed representations that preserve their contributions. This is particularly critical in SC vulnerability detection, where even subtle feature patterns

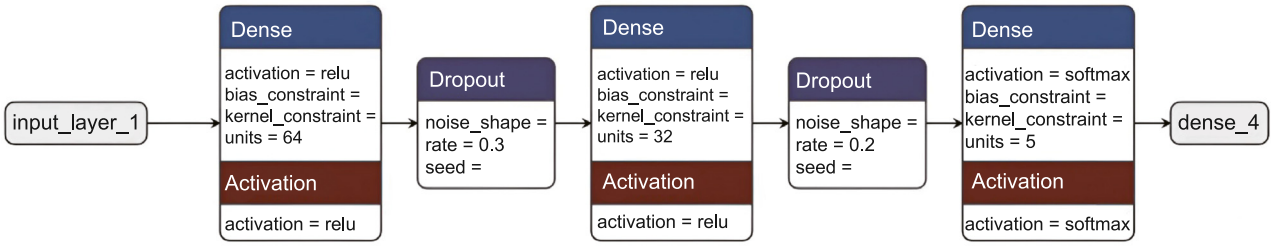

Fig. 7. Visual illustration of the utilized DNN for SC vulnerability detection.

may signal critical flaws. Moreover, feature selection techniques can become computationally expensive as the dataset grows in size and complexity, requiring iterative evaluation, ranking, or wrapper methods.

It is envisaged that new SC source codes may vary with time, making it necessary to update the configuration of the AE model concerning the streaming data. Therefore, the AE is proposed in the overall SC vulnerability architecture to dynamically incorporate new source code patterns. As new samples arrive, the AE continuously updates its learned representations, ensuring the model adapts to the changing SCs and maintains its detection capabilities. This dynamic adjustment is crucial for identifying unfamiliar patterns indicative of zero-day attacks and maintaining robust detection in dynamic SC vulnerability detection environments [56].

3.5. DL-enabled SC detection model

Next, this study leverages a simple yet high-learning Deep Neural Network (DNN) model to learn features from the SC features and categorize anomalous SCs from benign sources. This model was chosen for its ability to efficiently capture complex nonlinear relationships within SC feature representations while maintaining a lightweight architecture that supports fast training, strong generalization, and practical deployment in resource-constrained environments. The model architecture, as outlined in Eq. (6), and shown in Fig. 7, consists of a sequential neural network with three densely connected layers: an initial layer with 64 neurons, followed by a 32-neuron layer, all utilizing ReLU activation and interspersed with dropout regularization layers at rates of 0.3 and 0.2, respectively. The network concludes with a softmax activation output layer tailored for multi-class traffic classification, including zero-day attacks [57]. This architecture balances model complexity, generalization, and intrinsic learning ability [58] while avoiding the higher computational costs associated with more complex neural network models.

$$\hat{\mathbf{y}} = f(\mathbf{W}^{(L)} \cdot f(\mathbf{W}^{(L-1)} \cdot \ldots \cdot f(\mathbf{W}^{(1)} \cdot \mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) + \cdots + \mathbf{b}^{(L)}). \quad (6)$$

Here, $\mathbf{x}$ represents the vectorized network traffic input, $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the weight matrices and bias vectors for layer l, and f is the activation function applied element-wise—ReLU in this case. The output vector $\hat{\mathbf{y}}$ signifies the neural network's predictions. The model, compiled with the Adam optimizer (learning rate of 0.001) and categorical cross-entropy loss, incorporates EarlyStopping to monitor validation loss, with patience of 5 epochs and restoration of the best weights. Training was conducted over 50 epochs with a batch size of 32, and the total training time was recorded. This design effectively addresses the challenge of multi-class traffic classification while maintaining a balance between model complexity and computational efficiency.

3.6. Model-agnostic SC vulnerability detection with SHAP & LIME

Given the general taxonomy of AI model explainability as illustrated in Fig. 4 [59], some XAI tools are only favorable to only specific AI models and tasks. However, model-agnostic explanation methods outweigh model-specific methods in terms of explaining model decisions independent of its type and training data. In this study, the post-hoc model-agnostic tools employed are the SHAP [60] and LIME [61] explainers, which are tailored to provide model explanations regardless of the model type used for SC vulnerability detection. SHAP provides both local and global explanations, while LIME primarily offers local explanations. In the context of SC vulnerability detection, a local explanation provides insight into why the *black-box* SC detection model made a specific prediction for a single instance, by highlighting which input features contributed most to that individual decision. A global explanation, on the other hand, involves explaining why the SC vulnerability detection model made a specific prediction for a single instance by highlighting which input features contributed most to that individual decision.

The visual and quantitative cues that the model-agnostic explanations of SHAP and LIME provide help users and Blockchain-Metaverse experts or teams (those with a basic understanding of security and AI) understand the factors influencing the model's decision-making process. As represented in Eq. (7), LIME, with a good XAI visualization, does not make any assumptions about f, i.e., the model; it is only used to predict outcomes on new samples z acquired by altering x.

$$\arg\min L_{g \in G}(f, g, \pi_x) + \Omega(g). \quad (7)$$

Similarly, the SHAP explainer interprets the NIDS model structure based on many local explanations through defined intrinsic properties. The SHAP explainer is represented as follows in Eq. (8),

$$\pi(z') = \frac{(M-1)}{\binom{M}{z'}|z'|(M-|z'|)}, \quad (8)$$

where M is the maximum coalition size and z' is the number of present features in instance z'. Alongside security experts, users with basic visualization skills can validate security predictions based on the visual explanations provided. A summary of the overall approach proposed in this study is presented in Algorithm 1.

3.7. Dataset justification, description & preprocessing

The recent BCCC-SCsVul-2024 and BCCC-SCsVul-2023 datasets [55] were employed for the proposed SC vulnerability detection.[2] The authors of the dataset introduced BCCC-SCsVulLyzer

[2] https://www.yorku.ca/research/bccc/ucs-technical/cybersecurity-datasets-cds/

Table 1
SC dataset features Summary.

Feature category	Description
Contact information	Contract metadata, versions, and identifiers
Lines of code	Number of lines across different sections of code
Solidity features	Binary/categorical features for solidity functions and properties
Duplicate lines count	Number of repeated lines in the code
Event count	Total number of events defined in the contract
Functional features	High-level features describing the contract's functionality
AST features	Features based on the Abstract Syntax Tree (ABI), including length and node types
ABI features	Features describing the ABI, such as input/output types and mutability
Bytecode length and entropy	Bytecode length and its entropy measure for randomness
Opcode count features	Counts of various opcodes used in the bytecode
Bytecode character counts	Frequency of specific bytecode characters
Class features	Features representing potential vulnerabilities (e.g., gas exceptions, reentrancy)

Algorithm 1 Pseudocode of the proposed explainable metaverse SC vulnerability detection framework

Input: SCs $\mathcal{S} = \{s_i\}_{i=1}^{n}$, labels $\mathcal{Y}$, AE params θ_{AE}, DNN params θ_{DNN}, XAI methods $\mathcal{X} = \{\text{SHAP, LIME}\}$
1. Feature extraction & Encoding:
Extract features $\mathcal{F} = \{\mathbf{x}_i\}$ using SCsVulLyzer V2.0
Partition $\mathcal{F}$ into compiler- and non-compiler-derived features
Encode corresponding labels y_i, forming dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$
2. Dimensionality reduction (autoencoder):
Train autoencoder $\mathcal{A}_\theta = f_{\text{dec}}(f_{\text{enc}}(\cdot))$ to minimize:

$$\mathcal{L}_{AE} = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}_i - \mathcal{A}_\theta(\mathbf{x}_i)\|^2$$

Encode compressed representations $\mathbf{z}_i = f_{\text{enc}}(\mathbf{x}_i)$
3. Classification (DNN):
Train model $f_\theta : \mathbf{z}_i \rightarrow \hat{y}_i$ using cross-entropy loss
Apply regularization (e.g., dropout, early stopping)
4. Explainability (XAI):
for each $\mathbf{z}_i \in \mathcal{Z}_{\text{test}}$ **do**
Compute local/global attributions via $\mathcal{X}$
end for
Output: Predicted labels $\hat{\mathcal{Y}}$, latent features $\mathcal{Z}$, XAI explanations $\mathcal{E}$

-an open-source Python tool that profiles Ethereum SCs by extracting structural and behavioral SC features, such as entropy-based AST, ABIs, bytecode, opcodes, and source codes, to support statistical vulnerability detection and security analysis. This feature extraction tool maps raw codes into a meaningful feature space, enabling effective model classification.

The dataset comprises 253 features and 111,897 samples, capturing 11 distinct SC vulnerability classes, including External Bugs, Gas Exceptions, Mishandled Exceptions, Timestamps, and Transaction Order Dependence. For brevity, a summarized description of all 253 features is provided in Table 1. One of the features, *AST-Features-ast-nodetype*, which contains string labels representing AST node types in source units, is removed as it is very non-informative. Subsequently, columns with mixed data formats, those containing both (colon-separated triplets and float values), are identified. These columns are then parsed and split into separate numerical features: source, offset, and length from the triplets and a statistical value from the floats. The extracted components are concatenated into the data frame to support structured numerical analysis.

To facilitate streamlined model training, fine-grained threat categorization, and robust analysis, especially for underrepresented SC samples, the original multi-class dataset has been semantically grouped into five broader vulnerability categories based on domain knowledge [62,63], and as presented in Table 2. **Arithmetic & Gas Issues** (e.g., gas exceptions, integer overflows) pertain to computation errors and resource management flaws. **Control Flow & Logic Bugs** encompass issues such as mishandled exceptions and transaction ordering vulnerabilities, reflecting faults in logical execution and state transitions. **Access Control** groups vulnerabilities arising from improper permission handling and unauthorized function calls. **Runtime & Denial-of-Service (DoS)** includes threats compromising contract stability during execution, such as reentrancy and unhandled return values. Finally, the **Non-vulnerable** class comprises SCs with no identified security issues, serving as a baseline for comparison and classification.

4. Performance evaluation and analysis

4.1. SC detection model evaluation metrics

The proposed SC vulnerability detection model's performance is evaluated in terms of its effectiveness in classifying diverse, vulnerable SC types from benign codes. State-of-the-art performance metrics such as Accuracy (ACC) and Mathew Correlations Coefficient (MCC) scores are usually better for data imbalance, as well as the F1-Score (F1), Precision (PR), Recall (RL), Area Under the Receiver Operating Characteristic (AUC ROC), and resource consumption in terms of computational/training time (T). Additionally, we measure the XAI's confidence levels through benchmark quantitative prediction probability outputs of the SHAP and LIME explainers [64].

4.2. SC detection model evaluation metrics

The proposed SC vulnerability detection model is evaluated based on its effectiveness in distinguishing diverse vulnerable SC types from benign code. We report state-of-the-art performance metrics including ACC, MCC, F1, PR, RL, AUC ROC, as well as computational efficiency in terms of training and inference time (T). In addition, we assess the high or low confidence of predictions by calibrating and quantifying the probability outputs, and further analyze them through model-agnostic XAI explainers such as SHAP and LIME.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}, \tag{9}$$

where TP stands for True Positives, TN stands for True Negatives, FP stands for False Positives, and FN stands for False Negatives.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}, \tag{10}$$

$$\text{F1-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{11}$$

$$\text{Precision} = \frac{TP}{TP+FP}, \tag{12}$$

Table 2
Number of the BCCC-SCsVul-2024 SC samples per grouped vulnerability class and their severity within SC security.

Vulnerability group	Number of samples	Brief explanation
Non-vulnerable	26914	SC samples considered safe.
Arithmetic & Gas issues	23619	Potentially leads to overflow/underflow or out-of-gas errors, which can cause false logic executions.
Control flow & Logic bugs	14994	Can result in unexpected behaviors such as faulty business logic or unintended access to contract functions.
Access control issues	13049	High-risk; allows unauthorized users to execute privileged functions, potentially leading to asset theft.
Runtime & DOS	33321	Often exploited to halt contract functionality, lock user funds, and crash dApps via DoS attacks.

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{13}$$

The simulation for this study was conducted in a Python environment with the Tensorflow 2.13.0 library on a Windows 11 Pro, with the configuration of Intel(R) Core(TM) i7-14700KF CPU @ 3.40 GHz, 16 GB RAM, and GPU NVIDIA GeForce RTX 4060.

4.3. SC detection model performance

First, we analyze the performance of the SC detection model across the BCCC-SCsVul-2024 (five classes) and BCCC-SCsVul-2023 (binary class) benchmark datasets. The model performance of the two datasets is presented in Tables 3 and 4.

4.3.1. BCCC-SCsVul-2024 data scenario

From Table 3, we observe the following results. The SC detection model achieves an overall high precision, recall, F1-Score, accuracy, and MCC score of **97.74%, 97.73%, 97.73%, 97.13%, and 97.09%** respectively in a multi-class detection of *Non-Vulnerable, Arithmetic & Gas, Control Flow & Logic Access Control Issues, Runtime & DOS* attacks. Our focus on the high MCC score is particularly significant due to the MCC metric's robustness against class imbalance, as expected in real-world scenarios. The high MCC score suggests the model has strong predictive integrity even when some vulnerability types like (Control Flow & Logic Bugs, and Access Control Issues) are underrepresented.

Regarding class-wise detection performance, the SC detection model adequately captures the *Runtime & DOS* features owing to the strong source code entropy features of runtime activities and DoS attacks observed during the preprocessing stage. *Access Control Issues* yielded slightly lower recall (96.61%) but high precision (98.73%), suggesting the model is conservative in flagging such issues, potentially avoiding false alarms in sensitive contract areas. This slight gap in performance underscores the need for model verification with XAI. *Control Flow & Logic vulnerabilities* similarly showed a strong MCC score of 95.72%, highlighting the model's ability to capture logical flaws and control flow anomalies. In contrast, *Arithmetic & Gas-related vulnerabilities*, despite being structurally different from others, also maintained a high MCC score (95.67%). The class-wise performance confirms that the model generalizes well across diverse, vulnerable families. Furthermore, the multi-class results in Fig. 8(a) show the AUC-ROC curve for the predicted attack types. This visualization shows the detection model's impressive false positive rate in distinguishing attack instances with minimal false alarms—a critical attribute for SC vulnerability detection within the Metaverse.

As for model efficiency and real-world feasibility, the training and prediction time of **(95.52s and 0.050s)** as shown in Table 3 is justified by the higher class complexity and richer feature space required for multi-class discrimination. The low-latency prediction time results of the SC detection model show that the model using the BCCC-SCsVul-2024 dataset shows that the proposed detection framework is well-suited for real-time, latency-tolerant threshold (0.050s) auditing in automated SC pipelines. From a reliability and security standpoint, the Metaverse feeds on low-latency real-time predictions, especially in immersive marketplaces and decentralized applications. Thus, the execution time and model detection performance are well-suited for SC threat identification.

4.3.2. BCCC-SCsVul-2023 data scenario

As presented in Table 4, the SC detection model shows strong performance in binary classification of SCs as either *Secure* or *Vulnerable*, achieving an overall precision, recall, F1-Score, accuracy, and MCC of **94.56%, 94.37%, 94.43%, 94.37%, and 85.98%** respectively. The high overall F1-score and precision indicate the model's balanced capability to detect secure and vulnerable contracts with minimal performance trade-offs.

The MCC score of 85.98% supports the model's robustness and reliability, especially in real-world SC auditing environments. While the performance is slightly lower compared to the multi-class BCCC-SCsVul-2024 setup, this can be attributed to the fewer class distinctions in the 2023 dataset. Further class-wise analysis reveals the model performs equally on the *Secure* and the *Vulnerable* class with an equal MCC score of 85.98%. This performance suggests optimal prediction of vulnerable and malicious contract behavior in the Metaverse.

In terms of efficiency, the model demonstrates excellent runtime performance with a training time of **14.45s** and a prediction latency of **0.052s**, making it suitable for fast and lightweight deployment scenarios, complementing early-phase vulnerability detection, triage, and feedback when essential. Furthermore, as shown in 8, the multi-class AUC-ROC curve in 8(a) and binary AUC-ROC as demonstrated in Fig. 8(b) affirms the effectiveness of the model in distinguishing vulnerable contracts with acceptable generalization.

4.3.3. Additional validation on SmartBugs wild dataset

To further evaluate the effectiveness of our proposed XAI for SC vulnerability detection, we conducted additional experiments on the benchmark *SmartBugs Wild* dataset [36] containing 6057, 5958, and 5958 source codes, opcodes, and bytecode, respectively. It also has about 47,518 unique solidity files, encompassing 200,000 contracts. The model benchmark detection performance demonstrates adaptability to source code, byte code, and opcode-level inputs and confirms the practicability of our XAI framework beyond entropy-encoded datasets (BCCC-SCsVul-2024 and BCCC-SCsVul-2023). The core difference between the *BCCC-SCsVul* datasets and *SmartBugs Wild* training technique lies in its approach to text-based feature extraction. For the *SmartBugs Wild* dataset, the Term Frequency–Inverse Document Frequency (TF-IDF) vectorization technique is employed for feature extraction of Solidity SC codes. Afterward, an autoencoder is used to reduce dimensionality before training a DNN classifier, as presented in Fig. 7.

Based on the results shown in Table 5, the SC vulnerability detection model trained on the SmartBugs dataset demonstrates moderate but meaningful performance in detecting three key

Table 3
Performance metrics SC vulnerability detection model across five classes, using the BCCC-SCsVul-2024 dataset.

Class	Precision (%)	Recall (%)	F1-score (%)	Accuracy (%)	MCC (%)	Training time (s)	Prediction time (s)
Non-vulnerable	98.19	98.64	98.42	98.64	97.91	95.52	0.050
Arithmetic & Gas	96.83	96.34	96.58	96.34	95.67	95.52	0.050
Control flow & Logic	95.63	96.97	96.29	96.97	95.72	95.52	0.050
Access control issues	98.73	95.29	96.98	95.29	96.61	95.52	0.050
Runtime & DOS	98.58	99.29	98.94	99.29	98.49	95.52	0.050
Overall	**97.74**	**97.73**	**97.73**	**97.13**	**97.09**	**95.52**	**0.050**

Table 4
Performance metrics SC vulnerability detection model across five classes, using the BCCC-VulSCs-2023 dataset.

Class	Precision (%)	Recall (%)	F1-score (%)	Accuracy (%)	MCC (%)	Training time (s)	Prediction time (s)
Secure	97.35	94.91	96.12	94.37	85.98	14.45	0.052
Vulnerable	86.86	92.88	89.77	94.37	85.98	14.45	0.052
Overall	**94.56**	**94.37**	**94.43**	**94.37**	**85.98**	**14.45**	**0.052**

Table 5
Performance metrics SC vulnerability detection model across three classes, using the SmartBugsWild dataset.

Class	Precision(%)	Recall (%)	F1-score (%)	Accuracy (%)	MCC (%)	Training time (s)	Prediction time (s)
Clean	69.77	70.26	70.01	78.80	53.61	16.52	0.08
Reentrancy	63.82	62.22	63.01	76.07	45.34	16.52	0.08
Time Manipulation	70.63	71.91	71.26	81.44	57.56	16.52	0.08
Overall	**68.10**	**68.15**	**68.12**	**68.15**	**52.19**	**16.52**	**0.08**

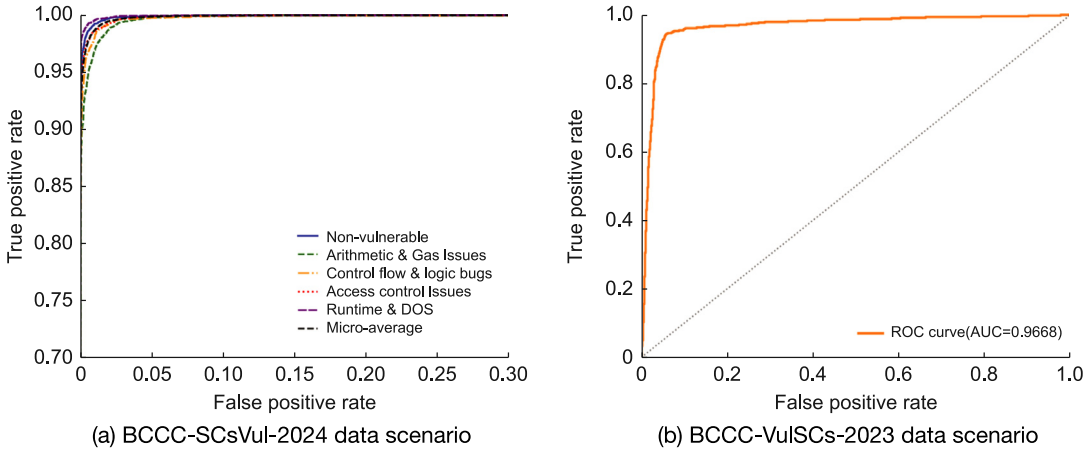

Fig. 8. (a-b) is the learning process curve showing the accuracy performance of the multiclass and binary class SC vulnerability classification using the bytecode entropy features of the BCCC-SCsVul-2024 and BCCC-SCsVul-2023 datasets.

vulnerability types: *Clean*, *Reentrancy*, and *Time Manipulation*. The model achieves an overall precision, recall, F1-score, accuracy, and MCC score of **68.10%, 68.15%, 68.12%, 68.15%, and 52.19%**, respectively. While these results are not as high as those observed on the previous datasets due to its homogeneous constituent of (source, bytecode, and opcodes), it still offer valuable insights into the model's detection integrity, especially when evaluated in real-world conditions. The moderate MCC of 52.19% underscores the challenge posed by the SmartBugs dataset, which inherently suffers from detecting diverse SC code vulnerability patterns that are difficult to distinguish without additional contextual features. Nevertheless, the MCC score still suggests a fair level of prediction reliability and is crucial in scenarios where class distribution skew could mislead simpler evaluation metrics.

For the *Clean* class, the model attains relatively strong detection metrics, with an F1-score of **70.01%** and the highest accuracy of **78.80%**. This indicates that the model is reasonably adept at differentiating non-vulnerable contracts, possibly due to their more consistent structural features. The *Time Manipulation* class exhibits the highest F1-score among the vulnerable categories at **71.26%** and the best MCC at **57.56%**, reflecting the model's ability to generalize over timestamp-related exploits. These vulnerabilities often manifest in specific real-world patterns, such as block timestamp or block number dependence, making learning easier for the model. On the other hand, *Reentrancy* vulnerabilities present a more significant detection challenge. With an F1-score of only **63.01%** and the lowest MCC of **45.34%**, the model demonstrates conservative precision (63.82%) but slightly weaker recall (62.22%). This disparity hints that the model may fail to capture complex reentrant fully flows, often requiring deeper semantic context or dynamic analysis features to be effectively flagged.

In terms of efficiency, the training and prediction times of **16.52s and 0.08s**, respectively, illustrate the lightweight nature of the model. Although slightly slower than real-time thresholds (compared to previous experiments), the low-latency prediction time still supports deployment in low-latency SC detection scenarios.

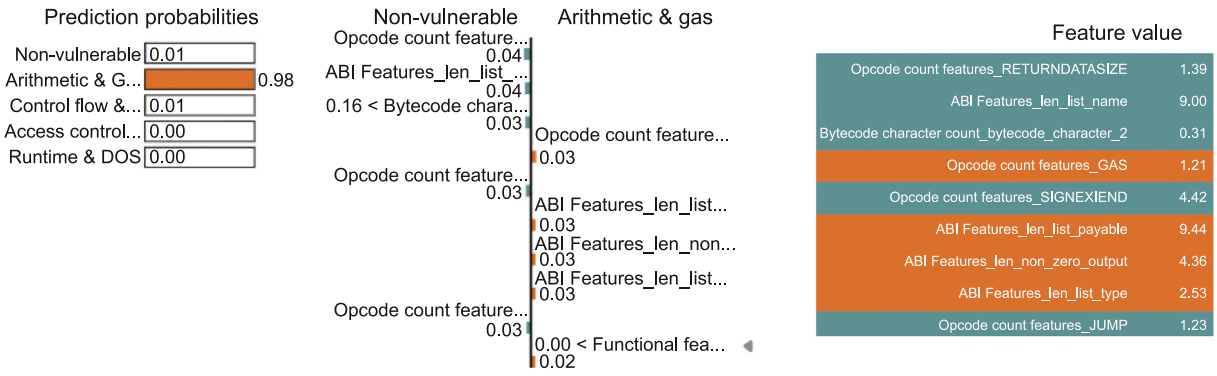

Fig. 9. LIME explanation for a high calibrated prediction score of 0.98 (arithmetic and gas consumption attack) for a vulnerable SC instance from the BCCC-SCsVul-2024 dataset.

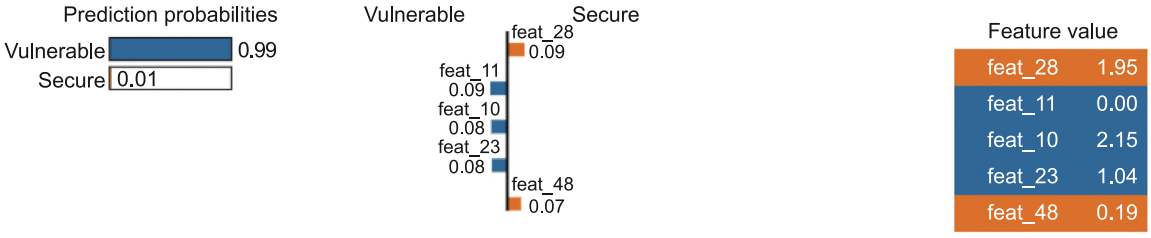

Fig. 10. LIME explanation for a high calibrated confidence prediction score of (0.99) for a vulnerable SC instance from the BCCC-SCsVul-2023 dataset.

4.4. XAI results

The visual XAI results, using model-agnostic explainers, employed for XAI-enabled Ethereum SC vulnerability detection, are discussed in this section.

These explainability results directly enhance the security and accountability of NFT reward issuance mechanisms in metaverse platforms. By providing interpretable insights into the model's decision process, auditors can verify the legitimacy of vulnerability classifications before executing automated reward functions. As demonstrated in Figs. 9 and 10, the entropy-based feature visualizations offer clear evidence of the model's confidence and feature relevance, thereby supporting traceable and trustworthy SC assessments. This interpretability is critical in preventing the misuse of automated NFT distribution, ensuring that only verified and non-malicious contract interactions trigger reward allocation. Such a capability is essential for maintaining integrity and user trust in Metaverse environments where NFTs often represent valuable digital assets or access privileges.

From Fig. 11, the SHAP explainer shows that the SC detection model can provide visual local interpretability for a random instance of a SC vulnerability class (*Access Control Issues*). Given the prediction output (f(x)) = 0.08, the model predicts a probability of 0.08 for the selected vulnerability class in the chosen test sample, which is close to the expected average base value (E[f(x)]) = 0.08.

The most influential and less influential features contributing to the SC detection model output are also captured using the model-agnostic SHAP explainer. As shown, opcode frequency features of the BCCC-SCsVul-2024 dataset, such as *Opcode Count Features-LOG3* and *Opcode Count Features-RETURN*, contribute positively to the prediction, while features like the *vulngroup* register the most significant negative SHAP value of −0.07, suggesting a substantial reduction in the predicted *Access Control Issues* probability when this feature is active.

SC Developers can make meaningful insights from the presented explanations for SC code refactoring [65]. Specifically, the security team can trace these SC features back to specific contract structures, bytecode segments, or opcode usage patterns. This deep-level diagnosis helps pinpoint risky coding practices, enabling better code reviews, testing, or rewrites of vulnerable logic (e.g., insecure *RETURN* behavior in access control). In essence, the XAI visualizations reveal which input features (e.g., entropy, opcode frequency) most influenced the model's decision, enabling developers to understand why a contract was classified as vulnerable. This XAI integration bridges the gap between model performance and helps SC auditors meet XAI transparency standards.

4.4.1. XAI results with raw features

In contrast to entropy-based features, visual XAI results from the raw bytecode, opcode, and source code features of the *Smartbugs Wild* dataset capture low-level and semantically rich representations of SCs, providing the model with detailed insights into control flow, memory access, and instruction sequences. A blockchain developer can utilize this LIME XAI in various insightful ways to enhance security, facilitate code refactoring, and meet the demands of trustworthy AI-enabled SC vulnerability detection. The developer reduces the time and effort required to read a larger number of raw codes, sees exactly why the model predicted a vulnerability, and identifies code patterns that tend to be risky.

Fig. 12 shows the LIME XAI library's prediction probability of the SC vulnerability detection model classifying a Solidity SC sample index as ***time-manipulation*** with high green bar confidence (91%). The remaining classes, such as reentrancy and clean classes, yield a very low prediction probability of 5% and 4%, respectively. Top contributing feature tokens, such as event, uint, public, VAR49, and address, appear frequently and are highlighted in green, indicating their role in pushing the prediction toward *time manipulation*.

Similarly, the LIME plot presented in Fig. 13 highlights the prediction probability for another random sample predicting a reentrancy attack with a 73% certainty. This time, the explainer struggles a bit as expected. However, based on the SC secu-

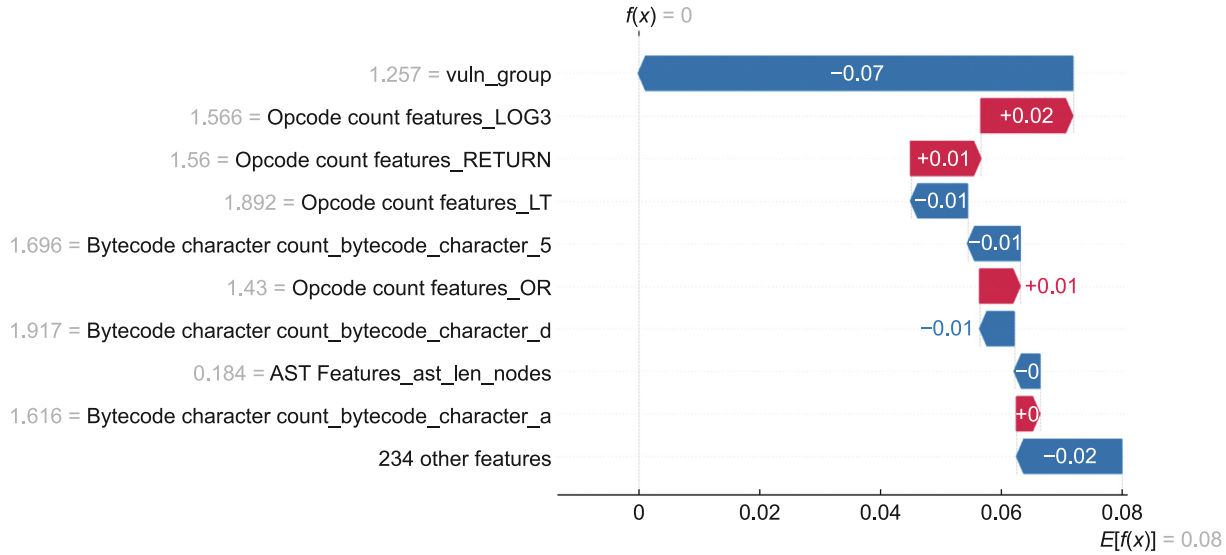

Fig. 11. SHAP waterfall plot illustrating the contribution of individual features to the DNN's predicted probability for a randomly selected test sample and target class.

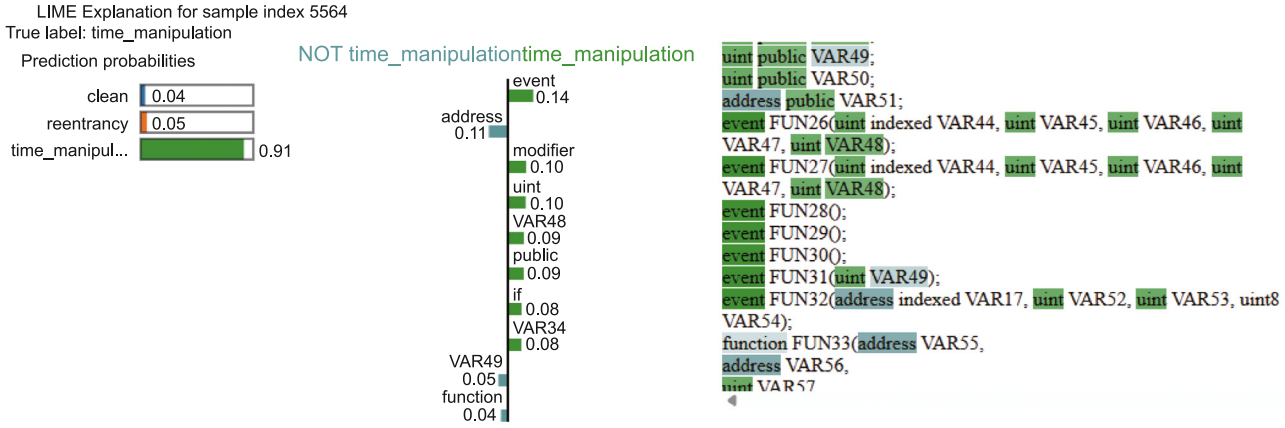

Fig. 12. LIME explanation for a high calibrated confidence prediction score of 0.91– *time manipulation attack* from the *Smartbugs Wild* dataset.

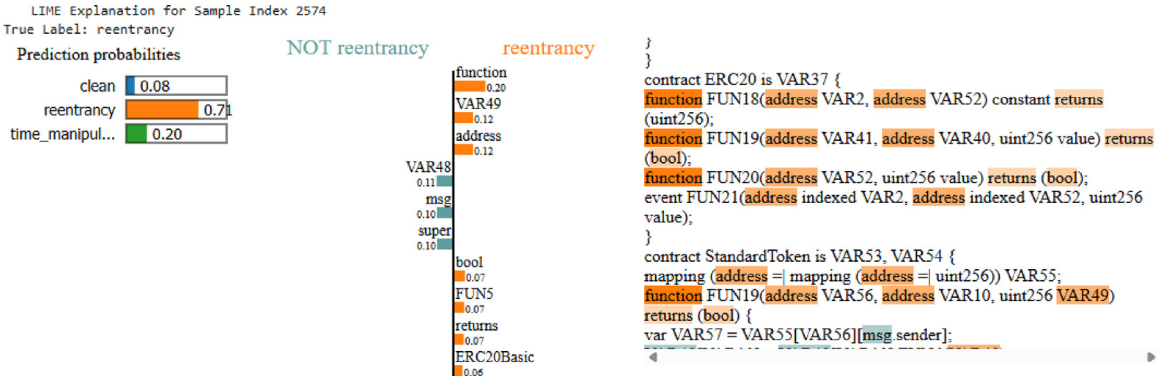

Fig. 13. LIME explanation with a high calibrated confidence prediction score of 0.71, showing a *reentrancy attack* from the *Smartbugs Wild dataset.*

rity viewpoint and the ground-truth vulnerability label for this contract, which is indeed reentrancy, the model's top prediction is correct and has been made explainable. Specifically, the orange-highlighted tokens (associated with reentrancy), including function, address, returns, VAR49, and ERC20Basic, are typically related to external function calls and value transfers, which

Table 6
Comparison table with related literature.

Author	Technique	Feature type				Dataset	Overall model accuracy	Model XAI
		source code	opcode	bytecode	control flow graph			
[26]	Static methods, RNN, LSTM, BiLSTM	✗	✗	✓	✗	Private	80	✗
[27]	CodeBERT, LSTM, CNN	✓	✗	✗	✗	SolidiFI	85.54	✗
[33]	GNN, BiLSTM, BERT	✓	✓	✗	✓	SolidiFI	77.96	✗
[36]	GNN	✓	✗	✗	✗	SmartbugsWild	95.49	✗
[66]	CodeBERT	✓	✓	✓	✗	SmartbugsWild	60.39	✗
Ours	**DNN**	✓	✓	✓	✗	**SmartbugsWild**	**68.15**	✓
[55]	CNN, DNN	**AST, ABIs, bytecode, opcodes, and source code**				BCCCL-SCsVul-2024 (12 classes)	–	✗
Ours	DNN	**AST, ABIs, bytecode, opcodes, and source code**				BCCCL-SCsVul-2024 (5 classes)	94.37	✓

✓: Supported; ✗: Not supported; –: Not applicable.

Table 7
List of abbreviations and their full meanings.

Abbreviation	Full meaning
AE	Autoencoder
AI	Artificial Intelligence
ABI	Application Binary Interface
AUC ROC	Area Under the Receiver Operating Characteristic Curve
CWE	Common Weakness Enumeration
DAO	Decentralized Autonomous Organization
DL	Deep Learning
DoS	Denial of Service
DNN	Deep Neural Network
F1-score	F1-score (harmonic mean of Precision and Recall)
FP	False Positives
FN	False Negatives
GNN	Graph Neural Network
GCNN	Graph Convolutional Neural Network
LIME	Local Interpretable Model-agnostic Explanations
LSTM	Long Short Term Memory
MCC	Matthew's Correlation Coefficient
ML	Machine Learning
PR	Precision
Recall	Recall
ReLU	Rectified Linear Unit
SC	Smart Contract
SHAP	SHapley Additive exPlanations
TP	True Positives
TN	True Negatives
XAI	Explainable Artificial Intelligence

are key indicators of a reentrancy attack, as described earlier in Section 1 and Fig. 1. This model-agnostic LIME visualization helps reveal the black-box's decision-making process, improving model transparency and trust in SC vulnerability detection.

4.5. Comparison with existing methods

Table 6 compares our method with existing state-of-the-art techniques in SC vulnerability detection. This comparison cuts across feature types, dataset usage, model accuracy, and explainability (XAI) support. A notable shortcoming of most existing works is the lack of interpretable and explainable support for detecting SC vulnerabilities. Our attempt is the first of its kind in this domain. Our framework integrates model-agnostic XAI methods to enable blockchain stakeholders to understand model decisions, a key aspect often lacking in related studies. We highlight that prior works such as [67] achieved higher accuracies under a binary classification setting (vulnerable vs. non-vulnerable contracts). In contrast, our multiclass formulation, distinguishing *clean, reentrancy, and time manipulation* contracts, thus presents a more complex task, which naturally leads to lower overall accuracy but provides finer-grained SC detection insights.

Furthermore, most prior work leverages SC's limited training features. For instance, [26,27] primarily utilize bytecode or source code alone. Our method is distinct in that it incorporates a diverse and rich set of features, ASTs, ABIs, bytecode, opcodes, and source code, allowing the model to capture syntactic and semantic nuances across different SC detection levels.

Compared to other approaches, the SC detection model achieves a competitive accuracy of 94.37% on the 5-class BCCC-SCsVul-2024 dataset and a better performance of 68.15% compared to benchmark methods (with an accuracy of %60.39) in [66] that have similarly employed the *Smartbugs Wild* dataset. Our work, in summary, demonstrates its utility and practicality for the proposed effective yet explainable SC detection framework.

Table 6 compares our method with existing state-of-the-art techniques in SC vulnerability detection. This comparison cuts across feature types, dataset usage, model accuracy, and explainability (XAI) support. A notable shortcoming of most existing works is the lack of interpretable and explainable support for detecting SC vulnerabilities. Our attempt is the first of its kind in this domain. Our framework integrates model-agnostic XAI methods to enable blockchain stakeholders to understand model decisions, a key aspect often lacking in related studies.

4.6. Discussion

In real-world scenarios, cryptocurrency compliance teams face global challenges ranging from technical limitations to legal constraints when auditing SCs deployed across decentralized finance (DeFi) platforms. At the same time, transparency in operations and financial reporting is critical for fostering trust between exchanges and end-users. Regulatory bodies such as the Markets in Crypto-Assets Regulation (Europe), Financial Conduct Authority (UK), Virtual Assets Regulatory Authority (Dubai), and the Office of Foreign Assets Control (USA) can benefit from XAI-enabled SC vulnerability detection frameworks to support forensic audits, dispute resolution, and court proceedings involving fraud or exploitation.

The results presented in this study advocate for model-agnostic explanation techniques that offer transparent, auditable insights into both vulnerable and secure SCs. Equally important, our proposed method introduces a notion of *high-confidence computing*, where calibrated predictive uncertainty and confidence quality metrics (e.g., quantitative model prediction probabilities) are used to quantify the reliability of model outputs. These quantitative signals enable compliance teams to set explicit thresholds for escalation, ensuring that low-confidence predictions are routed to human auditors or regulatory processes [68]. Such mechanisms strengthen regulatory oversight and operational transparency in high-risk financial environments.

Beyond traditional DeFi use cases, XAI-supported SC analysis has growing relevance in Metaverse marketplaces, where NFTs are used as tokenized representations of digital ownership, rights,

or rewards. Given the high value of NFT assets and their role in user engagement and monetization, ensuring that associated SCs are secure and trustworthy is critical. Confidence-calibrated explanations can be embedded into NFT issuance workflows, allowing security teams to distinguish high-assurance contracts from those that require additional review. This reduces the risk of exploits, fraud, or unintentional reward distribution due to undetected vulnerabilities.

Furthermore, XAI integration into user-facing platforms, such as blockchain wallets, NFT dashboards, or Metaverse asset explorers, can support end-user cybersecurity awareness. By exposing contract-level risks through interpretable visual explanations, together with confidence scores that reflect the reliability of predictions, non-technical users can make informed decisions before engaging in actions such as staking tokens, purchasing NFTs, or interacting with decentralized applications (dApps). This is especially important in immersive virtual environments where economic activity is fast-paced, user trust is paramount, and security teams must respond to threats in real time.

Overall, including interpretable and confidence-aware SC vulnerability detection frameworks in Metaverse infrastructures contributes to a more secure, transparent, and user-aware digital economy.

5. Conclusion and future direction

Compared to the extant literature in the SC vulnerability detection domain, whose focus has been on accuracy improvement, this paper introduces an explainable DL method for SC vulnerability detection. Based on experimental results, the efficacy of the proposed approach attains optimal detection performance when placed beside existing approaches. The major focus, model explainability, yields remarkable visual explanations with random entropy and raw source code features. The visual XAI results using model-agnostic explainers such as LIME and SHAP practicably bolster the confidence and trustworthiness of black-box DL methods employed for SC vulnerability detection. This research goes further to highlight how security teams, blockchain stakeholders, users, and compliance organizations can benefit from model-agnostic explainers to satisfy and improve resilient SCs (see Table 7).

Having established the need for XAI to enable trustworthy detection of vulnerable SCs, we are also aware of the vulnerability of DL- and LLM-enabled SC detection methods to adversarial and injection attacks. Also, Machine Unlearning has emerged as a new paradigm for an efficient and effective removal of vulnerable SCs from training pipelines. Therefore, our future work will address these gaps with an explainable and adversarially robust Machine Unlearning scheme for secure and privacy-preserving SC vulnerability detection.

CRediT authorship contribution statement

Ebuka Chinaechetam Nkoro: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Data curation, Conceptualization. **Love Allen Chijioke Ahakonye:** Writing – review & editing, Visualization, Validation, Supervision, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Dong-Seong Kim:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition, Formal analysis.

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.

Acknowledgments

This work was partly supported by Innovative Human Resource Development for Local Intellectualization program through the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (IITP-2025-RS-2020-II201612, 40%), by the Priority Research Centers Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology, South Korea (2018R1A6A1A03024003, 30%), and by the IITP (Institute of Information & Communications Technology Planning & Evaluation)-ITRC (Information Technology Research Center) grant funded by the Korea government (Ministry of Science and ICT)(IITP-2025-RS-2024-00438430, 30%).

References

[1] Y. Huang, Y.J. Li, Z. Cai, Security and privacy in metaverse: A comprehensive survey, Big Data Min. Anal. 6 (2) (2023) 234–247, http://dx.doi.org/10.26599/BDMA.2022.9020047.

[2] E.C. Nkoro, C.I. Nwakanma, J.-M. Lee, D.-S. Kim, Detecting cyberthreats in metaverse learning platforms using an explainable DNN, Internet Things 25 (2024) 101046.

[3] E.C. Nkoro, J.N. Njoku, C.I. Nwakanma, J.M. Lee, D.-S. Kim, MetaWatch: Trends, challenges, and future of network intrusion detection in the metaverse, IEEE Internet Things J. (2025).

[4] S.A. Gebreab, A. Musamih, H.R. Hasan, K. Salah, R. Jayaraman, Y. Al Hammadi, M. Omar, NFTs for accessing, monetizing, and teleporting digital twins and digital artifacts in the metaverse, Comput. Commun. 228 (2024) 107965, http://dx.doi.org/10.1016/j.comcom.2024.107965, URL https://www.sciencedirect.com/science/article/pii/S0140366424003128.

[5] H.R. Hasan, M. Madine, A. Musamih, R. Jayaraman, K. Salah, I. Yaqoob, M. Omar, Non-fungible tokens (NFTs) for digital twins in the industrial metaverse: Overview, use cases, and open challenges, Comput. Ind. Eng. 193 (2024) 110315, http://dx.doi.org/10.1016/j.cie.2024.110315, URL https://www.sciencedirect.com/science/article/pii/S0360835224004364.

[6] N. Andola, V.K. Yadav, S. Venkatesan, S. Verma, et al., Anonymity on blockchain based e-cash protocols—A survey, Comput. Sci. Rev. 40 (2021) 100394.

[7] S. Sayeed, H. Marco-Gisbert, T. Caira, Smart contract: Attacks and protections, *IEEE* Access 8 (2020) 24416–24427.

[8] K. O'Hara, Smart contracts - dumb idea, IEEE Internet Comput. 21 (2) (2017) 97–101, http://dx.doi.org/10.1109/MIC.2017.48.

[9] A. Nogueira, C.G. Marques, A. Manso, P. Almeida, NFTs and the danger of loss, Heritage 6 (7) (2023) 5410–5423.

[10] D. Krause, Crypto security in the aftermath of the bybit hack: Evaluating risk management strategies for digital assets, 2025, Available At SSRN 5156185.

[11] U. Agarwal, V. Rishiwal, S. Tanwar, M. Yadav, Blockchain and crypto forensics: Investigating crypto frauds, Int. J. Netw. Manage. 34 (2) (2024) e2255, URL https://onlinelibrary.wiley.com/doi/full/10.1002/nem.2255.

[12] G. Wu, H. Wang, X. Lai, M. Wang, D. He, S. Chan, A comprehensive survey of smart contract security: State of the art and research directions, J. Netw. Comput. Appl. 226 (2024) 103882, http://dx.doi.org/10.1016/j.jnca.2024.103882, URL https://www.sciencedirect.com/science/article/pii/S1084804524000596.

[13] Y. Wang, S. Sheng, Y. Wang, A systematic literature review on smart contract vulnerability detection by symbolic execution, in: International Conference on Blockchain and Trustworthy Systems, Springer, 2024, pp. 226–241.

[14] K. Li, Y. Xue, S. Chen, H. Liu, K. Sun, M. Hu, H. Wang, Y. Liu, Y. Chen, Static application security testing (SAST) tools for smart contracts: How far are we? Proc. the ACM Softw. Eng. 1 (FSE) (2024) 1447–1470.

[15] W. Gu, G. Wang, P. Li, G. Zhai, X. Li, Detecting unknown vulnerabilities in smart contracts with the CNN-bilstm model, Int. J. Inf. Secur. 24 (1) (2025) 33.

[16] V.K. Kasula, A. Reddy Yadulla, M. Yenugula, B. Konda, Enhancing smart contract vulnerability detection using graph-based deep learning approaches, in: 2024 International Conference on Integrated Intelligence and Communication Systems, ICIICS, 2024, pp. 1–6, http://dx.doi.org/10.1109/ICIICS63763.2024.10860016.

[17] V.K. Jain, M. Tripathi, An integrated deep learning model for ethereum smart contract vulnerability detection, Int. J. Inf. Secur. 23 (1) (2024) 557–575.

[18] H. Chu, P. Zhang, H. Dong, Y. Xiao, S. Ji, W. Li, A survey on smart contract vulnerabilities: Data sources, detection and repair, Inf. Softw. Technol. 159 (2023) 107221.

[19] I.H. Sarker, H. Janicke, A. Mohsin, A. Gill, L. Maglaras, Explainable AI for cybersecurity automation, intelligence and trustworthiness in digital twin: Methods, taxonomy, challenges and prospects, ICT Express (2024).

[20] N. Stephenson, Snow crash, Futures 26 (7) (1994) 798–800, URL https://doi.org/10.1016/0016-3287(94)90052-3.

[21] I. Mancuso, A.M. Petruzzelli, U. Panniello, Digital business model innovation in metaverse: How to approach virtual economy opportunities, Inf. Process. Manage. 60 (5) (2023) 103457.

[22] D. Vidal-Tomás, The new crypto niche: NFTs, play-to-earn, and metaverse tokens, Financ. Res. Lett. 47 (2022) 102742, http://dx.doi.org/10.1016/j.frl.2022.102742.

[23] S. Bhujel, Y. Rahulamathavan, A survey: Security, transparency, and scalability issues of nft's and its marketplaces, Sensors 22 (22) (2022) http://dx.doi.org/10.3390/s22228833, URL https://www.mdpi.com/1424-8220/22/22/8833.

[24] D. Das, P. Bose, N. Ruaro, C. Kruegel, G. Vigna, Understanding security issues in the NFT ecosystem, in: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 667–681, http://dx.doi.org/10.1145/3548606.3559342, URL https://doi.org/10.1145/3548606.3559342.

[25] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, A. Hobor, Making smart contracts smarter, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 254–269.

[26] T. Abdelaziz, A. Hobor, Smart learning to find dumb contracts, in: 32nd USENIX Security Symposium (USENIX Security 23), 2023, pp. 1775–1792.

[27] X. Tang, Y. Du, A. Lai, Z. Zhang, L. Shi, Deep learning-based solution for smart contract vulnerabilities detection, Sci. Rep. 13 (1) (2023) 20106.

[28] G. Crincoli, G. Iadarola, P.E. La Rocca, F. Martinelli, F. Mercaldo, A. Santone, Vulnerable smart contract detection by means of model checking, in: Proceedings of the Fourth ACM International Symposium on Blockchain and Secure Critical Infrastructure, BSCI '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 3–10, http://dx.doi.org/10.1145/3494106.3528672, URL https://doi.org/10.1145/3494106.3528672.

[29] C. Sendner, L. Petzi, J. Stang, A. Dmitrienko, Large-scale study of vulnerability scanners for ethereum smart contracts, in: 2024 IEEE Symposium on Security and Privacy, SP, IEEE, 2024, pp. 2273–2290.

[30] J.F. Ferreira, P. Cruz, T. Durieux, R. Abreu, Smartbugs: A framework to analyze solidity smart contracts, in: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering, 2020, pp. 1349–1352.

[31] L. He, X. Zhao, Y. Wang, Parse: Efficient detection of smart contract vulnerabilities via parallel and simplified symbolic execution, in: Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings, 2024, pp. 272–273.

[32] Y. Yao, H. Li, X. Yang, Y. Le, An improved vulnerability detection system of smart contracts based on symbolic execution, in: 2022 IEEE International Conference on Big Data (Big Data), IEEE, 2022, pp. 3225–3234.

[33] P.T. Duy, N.H. Khoa, N.H. Quyen, L.C. Trinh, V.T. Kien, T.M. Hoang, V.-H. Pham, Vulnsense: Efficient vulnerability detection in ethereum smart contracts by multimodal learning with graph neural network and language model, Int. J. Inf. Secur. 24 (1) (2025) 48.

[34] L. Guo, H. Huang, L. Zhao, P. Wang, S. Jiang, C. Su, Reentrancy vulnerability detection based on graph convolutional networks and expert patterns under subspace mapping, Comput. Secur. 142 (2024) 103894.

[35] N.A. Gupta, M. Bansal, S. Sharma, D. Mehrotra, M. Kakkar, Detection of vulnerabilities in blockchain smart contracts using deep learning, Wirel. Netw. 31 (1) (2025) 201–217.

[36] T. Durieux, J.F. Ferreira, R. Abreu, P. Cruz, Empirical review of automated analysis tools on 47,587 ethereum smart contracts, in: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, 2020, pp. 530–541.

[37] H. Chu, P. Zhang, H. Dong, Y. Xiao, S. Ji, W. Li, An integrated deep learning model for ethereum smart contract vulnerability detection, Int. J. Inf. Secur. 23 (2023) 557–575, http://dx.doi.org/10.1007/s10207-023-00752-5, URL https://link.springer.com/article/10.1007/s10207-023-00752-5#citeas.

[38] B.A. Juliussen, The right to an explanation under the GDPR and the AI act, in: International Conference on Multimedia Modeling, Springer, 2025, pp. 184–197.

[39] L.A.C. Ahakonye, C.I. Nwakanma, J.M. Lee, D.-S. Kim, Machine learning explainability for intrusion detection in the industrial internet of things, IEEE Internet Things Mag. 7 (3) (2024) 68–74, http://dx.doi.org/10.1109/IOTM.001.2300171.

[40] C.I. Nwakanma, L.A.C. Ahakonye, J.N. Njoku, J.C. Odirichukwu, S.A. Okolie, C. Uzondu, C.C. Ndubuisi Nweke, D.-S. Kim, Explainable artificial intelligence (XAI) for intrusion detection and mitigation in intelligent connected vehicles: A review, Appl. Sci. 13 (3) (2023) 1252, URL https://doi.org/10.3390/app13031252.

[41] E.C. Nkoro, C.I. Nwakanma, J.-M. Lee, D.-S. Kim, Detecting cyberthreats in metaverse learning platforms using an explainable DNN, Internet Things 25 (2024) 101046, http://dx.doi.org/10.1016/j.iot.2023.101046, URL https://www.sciencedirect.com/science/article/pii/S2542660523003694.

[42] F.S. Prity, M.S. Islam, E.H. Fahim, M.M. Hossain, S.H. Bhuiyan, M.A. Islam, M. Raquib, Machine learning-based cyber threat detection: An approach to malware detection and security with explainable AI insights, Human-Intelligent Syst. Integr. (2024) 1–30.

[43] S. Mahdavifar, M. Saqib, B.C.M. Fung, P. Charland, A. Walenstein, VulEXplaineR: XAI for vulnerability detection on assembly code, Mach. Learn. Knowl. Discov. Databases. Appl. Data Sci. Track (2024) 3–20.

[44] S.S. Taher, S.Y. Ameen, J.A. Ahmed, Advanced fraud detection in blockchain transactions: An ensemble learning and explainable AI approach, Eng. Technol. Appl. Sci. Res. 14 (1) (2024) 12822–12830.

[45] G. Thakur, S. Prajapat, D. Gautam, P. Kumar, G. Kumar, A.K. Bashir, H. Elmannai, A smart contract-based authentication scheme for securing metaverse environment in web 3.0, IEEE Trans. Consum. Electron. (2025) 1, http://dx.doi.org/10.1109/TCE.2025.3617939.

[46] R. Tewari, B.P. Pande, Transforming digital ownership: The role of NFTs in shaping virtual economies and market dynamics, IET Blockchain 5 (1) (2025) e70010, http://dx.doi.org/10.1049/blc2.70010.

[47] S. Su, Y. Tan, Y. Xue, C. Wang, H. Lu, Z. Tian, C. Shan, X. Du, Detecting smart contract project anomalies in metaverse, in: 2023 IEEE International Conference on Metaverse Computing, Networking and Applications (MetaCom), 2023, pp. 524–532, http://dx.doi.org/10.1109/MetaCom57706.2023.00095.

[48] J. Oppenlaender, The perception of smart contracts for governance of the metaverse, in: Proceedings of the 25th International Academic Mindtrek Conference, 2022, pp. 1–8.

[49] S. Banaeian Far, S.M. Hosseini Bamakan, NFT-based identity management in metaverses: challenges and opportunities, SN Appl. Sci. 5 (10) (2023) 260.

[50] H.X. Qin, Y. Wang, P. Hui, Identity, crimes, and law enforcement in the metaverse, Humanit. Soc. Sci. Commun. 12 (1) (2025) 1–15.

[51] M. Mosharraf, M.H. Khorrami, Creating non-fungible token (NFT)-backed emoji art from user conversations on blockchain, Data Sci. Manag. 8 (1) (2025) 40–47, http://dx.doi.org/10.1016/j.dsm.2024.06.002.

[52] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, B. Kim, Sanity checks for saliency maps, Adv. Neural Inf. Process. Syst. 31 (2018).

[53] T.R. Gadekallu, T. Huynh-The, W. Wang, G. Yenduri, P. Ranaweera, Q.-V. Pham, D.B. da Costa, M. Liyanage, Blockchain for the metaverse: A review, 2022, arXiv preprint arXiv:2203.09738.

[54] I. Mustafa, A. McGibney, S. Rea, An evaluation of lightweight CNNs for smart contract vulnerability detection, in: 2024 IEEE International Conference on Blockchain and Cryptocurrency, ICBC, 2024, pp. 675–677, http://dx.doi.org/10.1109/ICBC59979.2024.10634424.

[55] S. HajiHosseinKhani, A. Habibi Lashkari, A. Mizani Oskui, Unveiling smart contracts vulnerabilities: Toward profiling smart contracts vulnerabilities using enhanced genetic algorithm and generating benchmark dataset, Blockchain: Res. Appl. (2024) 100253, http://dx.doi.org/10.1016/j.bcra.2024.100253.

[56] A.A. Mohamed, A. Al-Saleh, S.K. Sharma, G.G. Tejani, Zero-day exploits detection with adaptive WavePCA-autoencoder (AWPA) adaptive hybrid exploit detection network (AHEDNet), Sci. Rep. 15 (1) (2025) 4036.

[57] V.T. Truong, L.B. Le, Metacids: Privacy-preserving collaborative intrusion detection for metaverse based on blockchain and online federated learning, IEEE Open J. Comput. Soc. (2023) http://dx.doi.org/10.1109/ojcs.2023.3312299.

[58] K. He, D.D. Kim, M.R. Asghar, Adversarial machine learning for network intrusion detection systems: A comprehensive survey, IEEE Commun. Surv. & Tutorials 25 (1) (2023) 538–566, http://dx.doi.org/10.1109/COMST.2022.3233793.

[59] G. Schwalbe, B. Finzel, A comprehensive taxonomy for explainable artificial intelligence: a systematic survey of surveys on methods and concepts, Data Min. Knowl. Discov. 38 (5) (2024) 3043–3101.

[60] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Adv. Neural Inf. Process. Syst. 30 (2017).

[61] M.T. Ribeiro, S. Singh, C. Guestrin, "Why should i trust you?" explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1135–1144.

[62] F.R. Vidal, N. Ivaki, N. Laranjeiro, Vulnerability detection techniques for smart contracts: A systematic literature review, J. Syst. Softw. 217 (2024) 112160, http://dx.doi.org/10.1016/j.jss.2024.112160, URL https://www.sciencedirect.com/science/article/pii/S016412122400205X.

[63] A. Singh, R.M. Parizi, Q. Zhang, K.-K.R. Choo, A. Dehghantanha, Blockchain smart contracts formalization: Approaches and challenges to address vulnerabilities, Comput. Secur. 88 (2020) 101654, http://dx.doi.org/10.1016/j.cose.2019.101654.

[64] C. Nicodeme, Build confidence and acceptance of AI-based decision support systems - explainable and liable AI, in: 2020 13th International Conference on Human System Interaction, HSI, 2020, pp. 20–23, http://dx.doi.org/10.1109/HSI49210.2020.9142668.
[65] M. Fowler, Refactoring: improving the design of existing code, Addison-Wesley Professional, 2018.
[66] Seaw1nd, Opcode-SmartContract, Kaggle (2024) URL https://www.kaggle.com/code/seaw1nd/opcode-smartcontract. (Accessed 24 April 2024).
[67] Y. Sun, S. Huang, G. Li, R. Chen, Y. Liu, Q. Jiang, A smart contract vulnerability detection method based on program dependency graph, in: 2023 4th International Conference on Computer, Big Data and Artificial Intelligence (ICCBD+AI), 2023, pp. 84–89, http://dx.doi.org/10.1109/ICCBD-AI62252.2023.00022.
[68] L. Bellantuono, F. Palmisano, N. Amoroso, A. Monaco, V. Peragine, R. Bellotti, Detecting the socio-economic drivers of confidence in government with explainable artificial intelligence, Sci. Rep. 13 (1) (2023) 839.