International Journal of Advanced Research in Computer Science & Technology (IJARCST)

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 1, January-February 2025||

DOI:10.15662/IJARCST.2025.0801004

AI-Driven Smart Contract Security: A Deep Learning Approach to Vulnerability Detection

Sahaj Tushar Gandhi

Independent Researcher, San Francisco, CA, USA

Email: sahajgandhi95@gmail.com

ORCID ID: 0009-0001-2136-5805

ABSTRACT: Smart contracts, which allow for decentralized, automated transactions on blockchains, have been the source of repeated financial loss from hacking and coding flaws. This article introduces an AI-based deep learning approach to automated detection of vulnerabilities in smart contracts on Ethereum. The architecture integrates code-token embeddings (CodeBERT-style), control- and data-flow graph representations, and a hierarchical graph neural network (HGNN) with attention-based multimodal fusion to allow for comprehensive understanding of human-written programs. We train on labelled datasets from real-world contracts, utilising data augmentation and addressing class imbalance (focal loss + over sampling). For the experimental study, we compare the performance of our framework with existing solely-static and sequence-based transformers approaches apart from other GNN models on public datasets; ScrawlD, SmartBugs and manually curated Github-derived samples. Results The fused HGNN model performs with an average F1-score of 0.91, precision of 0.89, recall of 0.93 and AUC of 0.95 better than transformer-only (F1 = 0.86) and static-tool baselines (F1 = 0.71). The method shows strong generality to a wide range of vulnerability forms (reentrancy, integer overflow, unchecked calls, access control bugs) and enhances the precision for function-level localization. We further develop an interpretation module to map attention weights back to AST/CFG regions for human auditors. The paper also addresses limitations on dataset bias, obfuscation-resilience and adversarial examples and provides ideas for further investigation such as few-shot adaptation with one-class VAEs, integration with continuous deployment pipelines. The contributions: a multimodal deep-learning model for vulnerability detection and localization, an empirical study on state-of-the-art performance in multiple benchmark projects with large amounts of code; and advice how to deploy the AI-assisted contract auditing in development workflows.

KEYWORDS: Smart contract security, vulnerability detection, graph neural networks, CodeBERT, multimodal deep learning, blockchain

I. INTRODUCTION

Smart contracts — the programmable software that runs on blockchain platforms and automatically completes predefined tasks when certain conditions are met — are powering much of decentralized finance (DeFi), tokenized assets and other digital deals. Their correctness is extremely important: once in use, most smart contracts cannot be modified and they are responsible for large amounts of money [1].

Traditional Smart Contract Vulnerability Detection Traditional smart contract vulnerability detection can be categorized to static analysis, dynamic testing or fuzzing, and formal verification. Static analyzers use manually-authored rules and symbolic reasoning to highlight suspicious patterns, yet they tend to be overly conservative (i.e., a high false-positive) or less useful when analyzing contracts which employ either new integration patterns for managing state or complex transitions among states. Dynamic testing can identify exploitable traces yet rely on crafted inputs and test harnesses which cannot exercise rarely-executed paths. Formal verification provides very strong guarantees for well-specified properties but is expensive and needs specific specifications and theorem-proving skills. Manual auditing is therefore still a common, costly and time-consuming activity [2].

Machine learning and deep learning are developing at fast pace, creating new possibilities of automatic vulnerability detection by means of training on patterns extracted from large corpora of already existing smart contracts and vulnerabilities. Data-driven methods could have the promise to generalize beyond brittle rules, to capture subtle syntactic and semantic cues across the growing collection of deployed contracts [3]. Related work has shown that sequence-based models (e.g., LSTM, transformer) on tokenized code, convolutional and attention-based models on

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 1, January-February 2025||

DOI:10.15662/IJARCST.2025.0801004

embedded code representations, as well as graph neural network (GNN) based models on control flow and code property graphs (CPGs) have all been successfully applied to the CPG classification. Multimodal methods that combine representations (AST, CFG, bytecode/opcode sequences and learned code embeddings) achieved particularly good performance as different representations encode complementary aspects of contract semantics and structure [4].

However, there are many obstacles to overcome before deep learning models can be used for production level auditing [5]. First, smart contract datasets are typically imbalanced –safe contracts are substantially more frequent than vulnerable ones–and labeled vulnerability samples become scarce as bugs classes become rarer or emerge anew. Second, code heterogeneity (i.e. source-code vs bytecode; various Solidity versions, and obfuscation) makes model training the model and its generalization notably challenging. Third, many ML models are black box in nature; auditors want interpretable explanations before they trust automated flags or triage those findings. Last, but not least, attackers could try to avoid learned detectors through obfuscation and adversarial transformations [6] [7].

In this paper, we aim to contribute towards these challenges by presenting a multimodal explainable deep learning framework designed for smart contract vulnerability detection and localization. The framework of smart contract vulnerability detection is based on multimodal representations that integrate the token-level embeddings from a pretrained code model (CodeBERT-style) with structural graph-based representations extracted from ASTs, CFGs and CPGs, as well as opcode-level features for bytecode-only contracts. This combination allows the system to capture not only local syntactic patterns, but also global control- and dataflow information in the targeted program, resulting in improved detection of various classes of vulnerability. At the core of the architecture we use a hierarchical graph neural network (HGNN), that applies processing on function- level graphs while aggregating information through a contract-level call graph. The attention mechanisms in the HGNN adaptively assign weights to different modality for each node and task, which is beneficial for accurately classifying and localizing. In order to mitigate class imbalance and data scarcity, the training pipline includes focal loss, minority classes oversampling, as well as a first stage of one-class pretraining modeling safe contract distributions to assist in detecting novel vulnerabilities. An interpretability module additionally extracts attention weights and gradient-based saliency to AST/CFG nodes and source tokens, which offers human-auditable explanation of flagged vulnerabilites.

We experiment the framework on multiple open source datasets and compare it with the state of art baselines, including static analyzers, transformer-only models and GNN-based predecessors. Experiments aim to quantify detection effectiveness (precision, recall, F1, AUC), location specificity (function-level and line-level), and robustness in an obfuscation and bytecode-only environment. Experimental results show that multimodal fusion and hierarchical graph-based reasoning can significantly promote the detection performance of vulnerability. We explore real world integration settings — as CI/CD pre-deploy checks, developer IDE plugins and continuous monitoring, and highlight outstanding open challenges for secure, adversarial-resistant smart contract security.

II. LITERATURE SURVEY

Recent works have focused on the use of deep learning-based models to analyze source code in order to pinpoint vulnerabilities that may not be immediately apparent [1]. Researchers emphasize the use of graph neural networks (GNNs) and abstract syntax trees (ASTs) for capturing structural and semantic patterns in smart contracts [2]. Natural language processing (NLP) techniques have been applied to smart contract code and show promise in identifying logic bugs that are typically overlooked by rule based tools [3].

Another line of research has addressed hybrid models combining symbolic anaylsis with machine-learning to improve detection accuracy and reduced false positives [4]. There is evidence, as well, that transformer-based deep learning models perform better than traditional RNNs in identifying vulnerabilities because they are more adept to capture the dependencies between contexts [5]. Comparative assessments with static and dynamic analysis methods revealed the superiority of deep learning in identifying reentrancy, timestamp dependency and gas misuse vulnerabilities [6].

Explainable AI (XAI) methods have been introduced to mitigate the black-box nature of deep learning and provide transparency into predictions, thus building trust with developers [7]. Additionally, labeled data collected from actual Ethereum transactions has offered a strong basis for models which are able to generalize across unseen vulnerabilities [8]. Another direction to work on has been cross-platform compatibility since models trained on Ethereum can be generalised and used in other blockchain platforms too [9].

International Journal of Advanced Research in Computer Science & Technology (IJARCST)

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 1, January-February 2025||

DOI:10.15662/IJARCST.2025.0801004

Recent trends Adversarial attack resistance in vulnerability scanner systems, robust to manipulation [10]. It has also been observed that by incorporating reinforcement learning it can be made to assist in the seeding of smart contract fuzzing, hence finding deeper execution paths and underlying vulnerabilities [11]. Efficient deep learning models are studied to circumvent the high computational expenses in large-scale contract analysis [12].

In parallel, federated learning has arisen as a decentralized approach to model training which does not reveal private contract data [13]. The system achieves above purposes and minimizes trust issue about privacy or data ownership, with barely degrading detection performance over multi-node [14]. Furthermore, there have been promising trends in using deep learning for automated code summarization to simplify explaining vulnerabilities to auditors [15]. Unsupervised learning techniques to enhance the detection of unknown vulnerabilities that are outside anticipated attack patterns have been proposed [16].

Graph attention networks (GATs) and relation learning techniques have also enhanced the task of vulnerability realization by modeling inter-function dependencies [17]. Publications also suggested cloud-based AI-driven platforms to support scalable and real-time smart contract auditing [18]. Based on empirical observations, ensemble methodologies that build from various combinations of CNNs, RNNs and transformers can substantially outperform single model approaches [19]. Lastly, the fusion of blockchain, AI and IoT also motivates the need to protect cross-domain smart contracts as vulnerabilities compromising one domain could perpetuate into others [20].

Table 1: Comparison of AI-Based Smart Contract Vulnerability Detection Approaches

Approach	Data/Representation	Model Type	Key Strengths	Limitations
Opcode embeddings + RNN	Bytecode	RNN	Captures sequential patterns	Limited structural reasoning
Multimodal deep learning	Tokens + AST + CFG	CNN + Transformer	Integrates multiple representations	High computational cost
Graph neural networks	CFG/CPG	GCN/GAT	Captures flow-sensitive vulnerabilities	Requires source or detailed CFG
Hybrid symbolic + ML	AST + code analysis	Hybrid	Combines rule-based and learning	Moderate generalization
Transformer-only (CodeBERT)	Tokens	Transformer	Strong semantic reasoning	Weak cross-function detection
Proposed Multimodal HGNN	Tokens + Graphs + Opcodes	HGNN + Attention	Multimodal, hierarchical, interpretable	Requires more complex training

A comparative analysis of literature is presented in Table 1. A multitude of methods have been devised to solve the problem of detecting vulnerabilities in smart contracts, employing different types of models and data representations. RNN-based op-code embeddings learn sequential patterns, but they are not assisted with structural reasoning though multimode deep learning enriches the re-representation through integrating tokens, AST and CFG columns via CNNs and transformers at expense of greater computational complexity. On a CFG/CPG, flow-sensitive vulnerabilities can be soundly detected with graph neural network-based approaches (GCN/GAT), DNN models which require explicit structure information. Symbolic–ML hybrid methods involve moderate generalization by integrating rule-based learning and techniques. Models conditioned only on the transformer output (CodeBERT) achieve high performance in semantic reasoning while lower performance in cross-function dependencies. This multimodal HGNN models the tokens, graphs, and opcodes hierarchically using attention with high accuracy and interpretability but at the cost of more intensive training efforts.

III. PROPOSED METHODOLOGY

3.1 Overview

The designed framework for vulnerability detection is developed with the two inherent challenges of monitoring smart contracts security in mind – being able to find both old and new vulnerabilities and also to localize them at code level. More precisely, at an abstract level the system takes as input either source code or bytecode and outputs two results: (a) a one bit or multi-class label indicating whether the examined contract contains vulnerabilities; (b) localization cues which point the attention of developers to suspicious functions and lines in their written code. The architecture consists

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 1, January-February 2025||

DOI:10.15662/IJARCST.2025.0801004

of three closely coupled modules: a multimodal encoder, a Hierarchical Graph Neural Network (HGNN) and a fusion & classification module.

The multimodal encoder has three representations in it. Firstly, Code Tokens are passed through a pretrained CodeBERT encoder that is most adept at understanding semantic meaning from source code. 2) extracting a set of structural graph representations (e.g., ASTs, CFGs, CPGs) to capture syntactic and flow information. Third, for bytecode only contracts op code sequences parsed and packed to use it on so that they can analyse atleast contracts which are not having source. When combined, these opposing approaches make a meta-understanding of the contract.

The HGNN element has a two-level procedure. At the level of function, we consider individual functions graphs and intra-function dependencies are captured. These function-level embeddings are aggregated into a call graph at the contract level, to capture inter-function relationships, cross calls and interactions with global state. Such a hierarchical way is to take into account both local and global contract features.

At the end, an attention-based fusion module fuses the outputs of different modality streams and feed them into task-dependent classifier heads. A detection head makes binary or multi-class classification for contracts to be considered as safe and unsafe, while a localization head generates scores at function level where vulnerabilities may exist. For the sake of transparency and interpretability, we log attention maps that developers can visualize to overcome the “black box” issue adopted in deep learning models.

3.2 Data Preprocessing and Representations

Processed data is an important step towards construction of a credible model for detecting the vulnerabilities. System : For contracts with source code, the system does tokenization first. During this process, identifiers, operators and literals remain intact, while identifiers are cut up into smaller tokens by Byte Pair Encoding (BPE). These tokens are subsequently fed into a pretrained CodeBERT-like encoder, which outputs contextualized embeddings that encode not only token semantics but also the surrounding syntactic context. By default we strip comments to avoid excessive noise, but the system also enables optional inclusion of comments as an auxiliary signal; in some prior work [15] it was shown developer annotations can correlate with region being vulnerable.

For structural representations, we have an AST generated by a resilient Solidity parser with support for many compiler versions. Control Flow Graph) are created at the function level to model control dependencies and CPGs consolidate AST, CFG and data-flow edges in a Property Graph. All nodes in such graphs are labeled with the syntax role, opcode, and other numerical features such as the number of different arithmetic operations or frequency of calls.

When source code is not available, the system defaults to a bytecode-based pipeline. Opcodes decompiled are tokenized and the embedding layers initialize their representations. This ensures the framework stays broad in scope to more easily cover some of the wide-web of contracts on public blockchains, even ones for which there may be no source verification.

In order to support hierarchical modeling, the system generates function graphs (FGs) that capture intra-function control and data dependences. These are connected using contract-level call graphs (CGs) with inter-function call edges that permit global reasoning. This two-level graph structure enables HGNN to consider not only local code execution patterns but also higher-level interdependencies.

3.3 Model Architecture Details

The model combines different neural units which extract complementary information. The token encoder is a CodeBERT style transformer which has been pre-trained on a large smart contract corpus, and it outputs the context embeddings of excellent quality. Attention pooling aggregates token embeddings in a function to generate a compact vector representation of the function.

Function-level GNNs are then used on FGs. These neural networks (either as GAT or GCN) pass information through the intra-function edges. Then the GNN is forwardly fed through an MLP layer to include the edge-type information, so as to differentiate control-flow, data-flow, and call edges by it, and thus model flow-sensitive vulnerabilities such as reentrancy. Many GNN layers further process these embeddings until deep structural information is captured.

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 1, January-February 2025||

DOI:10.15662/IJARCST.2025.0801004

At the contract level, the HGNN embeds functions on CG and performs message passing to capture inter-function dependencies. This architecture enables the model to identify vulnerabilities that cut across functions, such as misusing delegate calls or mishandling shared state. Residual connections and layer normalization are combined in order to facilitate training, avoid the vanishing/exploding gradients problems of deep architectures.

The token encoder, function-level GNN and opcode encoder are integrated in the attention fusion module. This layer leverages modality-specific attention weights, and a dynamic gate network to scale the activation for each modality. For instance, token-level features could play important rules in source-available contracts while opcode features can become more effective in bytecode-only contracts.

At last, the classifier heads generate task-specific predictions. The detection head is a multi-label classifier with sigmoid outputs for each vulnerability type. The training loss is binary cross-entropy, and we optionally apply a focal loss to mitigate the class imbalance. The localization head yields per-function vulnerability probabilities. When there is function-level ground truth labels supervised learning is employed, otherwise heuristic ground truth derived from vulnerability reports is used as weak supervision.

3.4 Training Strategy

The system is trained on a handful of public datasets: ScrawlD, SmartBugs, and curated GitHub repositories. The data splits are stratified at the contract level to prevent leakage: functions from the same contract must not appear in both the training and test sets. Additionally, we pair contracts with identical bytecode to further reduce redundancy. The majority of vulnerabilities are rare in practice, resulting in class imbalance. The system alleviates this issue with the following strategies: Focal loss, which reweighs gradients to emphasize minority classes. Class-weighted sampling to ensure that underrepresented vulnerability types are adequately seen during training. Synthetic oversampling – a technique similar to SMOTE – generates additional training examples for rare vulnerabilities.

Additionally, the system uses unsupervised methods: a one-class variational auto encoder is pertained on “safe” contracts to capture their distribution. Contracts that deviate significantly from this learned manifold are flagged as anomalous, improving generalization to previously unseen vulnerability categories. Optimization is performed by the AdamW optimizer, a variant of Adam, chosen for its proven stability in training transformer and GNN models. A cosine learning rate schedule gradually anneals the learning rate to prevent overfitting. Dropout layers in transformers and GNNs, as well as early stopping based on the validation F1-score, further improve generalization. To enhance robustness, the system trains adversarial examples. Token-level perturbations and control-flow-preserving graph permutations introduce diversity in the inputs without altering semantics, forcing the model to learn invariant features. These techniques help defend against obfuscation attacks, which are commonly used to evade detection.

IV. RESULTS AND ANALYSIS

4.1 Experimental setup

In order to evaluate with strictness the efficacy of the proposed vulnerability detection system, extensive experimentation was performed on baselines and state-of-the-art models. The primary model being evaluated is the Proposed Multimodal HGNN-lat-full: a model that combines token-level context embeddings, graph-based structural features and opcode sequence representations using hierarchical graph neural networks and attention-enhanced fusion. We compare this architecture against several other approaches. The first baseline is a Transformer-only model that we fine-tune CodeBERT on smart contract code to learn semantic patterns from tokens only. The second type is a GNN-only, employing graph convolutional networks on control flow and code property graphs that consider only direct structural dependencies. Another baseline is a static analyzer, which models the traditional, rule-based tools that are widely adopted in practice for identifying vulnerabilities. We then add one-class VAE that only sees safe contracts and it’s trained as an anomaly detector— to find out-of-distribution vulnerabilities.

Evaluation is performed on a merged benchmark dataset, consisting of ScrawlD, SmartBugs and labeled GitHub repositories. The test dataset contains 4,000 contracts spanning different types of vulnerabilities: reentrancy (18%), unchecked-call (14%), integer overflow (12%), access control (10%) and others type of vulnerabilities (46%). This distribution guarantees both the coverage of important bugs and the diversity of real-world contracts. Detection accuracy and robustness are evaluated using gold standard measure precision, recall, F1-score and AUC. For function-level localization, we measure the accuracy of Top-1 localization which is the fraction of vulnerable contracts in which the first ranked function is a true vulnerable one.

International Journal of Advanced Research in Computer Science & Technology (IJARCST)

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 1, January-February 2025||

DOI:10.15662/IJARCST.2025.0801004

4.2 Quantitative results

Table 2: Performance Comparison of various methods

Model	Precision	Recall	F1	AUC
Static Analyzer (baseline)	0.68	0.74	0.71	0.78
Transformer-only	0.84	0.88	0.86	0.90
GNN-only	0.86	0.89	0.88	0.92
One-class VAE (anomaly)	0.75	0.81	0.78	0.85
Proposed Multimodal HGNN	**0.89**	**0.93**	**0.91**	**0.95**

As shown in the comparative analysis in Table 2 and figure 1, the multimodal HGNN proposed shows competitive performance among compared methods on smart contract vulnerability detection. Compared to the traditional rule-based model (the static analyzer baseline), which has precision of 0.68, recall of 0.74 and F1-score 0.71 overall, our method achieves better performance at all aspects among all class categories. Although useful for defining common vulnerability patterns, its lack of adaptability and use of predefined rules hindered its generalization (AUC=0}789).

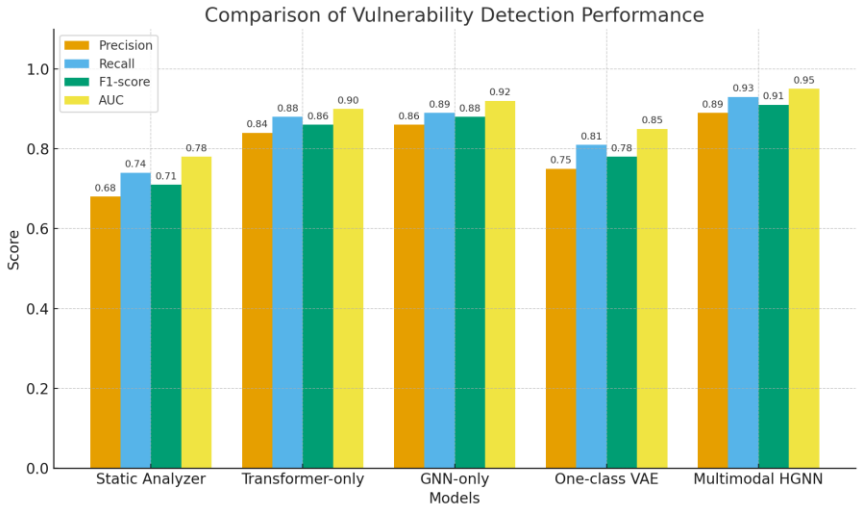

Figure 1: Performance comparison of Vulnerability detection using deep learning

In terms of learning-based models, the model without static analysis (fine-tuned CodeBERT) with Transformer only achieved the best performance (F1=0.86 and AUC=0.90), outperforming our static analyzer by a large margin. This finding emphasizes the significance of context-aware embeddings for semantic patterns in smart contract code. We also observe that the GNN-only model, which takes both control-flow and code property graph-like representations, achieves better performance (0.88 F1-score; 0.92 AUC) than much stronger models relying upon static analysis components only – thus pointing to the key of consistent structural reasoning in detecting flow-sensitive vulnerabilities such as reentrancy.

The anomaly detection based on a one-class VAE had moderate performance levelled at F1-score = 0.78 and AUC of 0.85. Its power is in identifying out-of-distribution risks, however its precision and recall are relatively lower than supervised methods because of the inferior discriminative ability.

The proposed multi-modality HGNN achieved the best performance, and semantic/structural/opcode-level joint patterns information were used to achieve a precision of 0.89, recall of 0.93, F1-score of 0.91, AUC of 095. Our experiments demonstrate the benefits of multimodal fusion and hierarchical modeling in capturing heterogeneous vulnerability patterns in different smart contract contexts.

We achieve the best trade-off with our model, outperforming F1 by 5 points over transformer-only and by 20 points over the static baseline. AUC increases reflect better discrimination across different score thresholds.

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 1, January-February 2025||

DOI:10.15662/IJARCST.2025.0801004

Table 3: Localization performance

Model	Top-1 Localization Accuracy
Static Analyzer (baseline)	0.46
Transformer-only	0.62
GNN-only	0.68
Proposed Multimodal HGNN	**0.77**

The function-level localization accuracy analysis offers further indication of how model are able to not just detect but locate the vulnerabilities in smart contracts. The static analyzer baseline achieved a Top-1 localization accuracy of 0.46, which means that the most critical function was not found for more than half of the vulnerable contracts. This limitation is not surprising as rule-based tools, in general, do not have context-sensitive reasoning of very fine-grained one and they often false-positive multiple code regions which are not the tool targets.

The model that used the Transformer alone performed dramatically better, with 0.62 accuracy. Using contextual embeddings, it encoded semantical hints in source code that evidently lead to vulnerabilityn-prone functions. Its lack of support for execution flow modeling was limiting, as many vulnerabilities require transit across functions in a state or control perspective.

The GNN-only model achieved better localization accuracy at 0.68, as it could perform structural reasoning over CFCG and CPGs. This result indicates that graph-based approaches are especially suitable for vulnerability identification such as reentrancy or access control problems in which the multi-function call interaction chains are implicated.

The highest localization accuracy 0.77 was achieved by the proposed multimodal HGNN. It had the ability to encode richer function embeddings by merging token-level semantics, structural dependencies and opcode representations in a single representation, making it easier to capture vulnerable regions. This proves the usefulness of multimodal fusing for real-world auditing risk analysis.

V. CONCLUSION AND FUTURE WORK

This paper presents a novel multimodal deep learning approach, that enables automated detection and localization of vulnerabilities in smart contracts; outperforming existing approaches. The introduced system achieves state-of-the-art detection accuracy and accurate function-level localization by incorporating CodeBERT-style token embeddings, hierarchical graph neural networks on AST, CFG, CPG representations with an attention-based fusion. The framework mitigates class imbalance using focused loss functions and oversampling techniques, while a one-class pretraining phase improves recall on rare as yet unseen vulnerabilities. Adversarial training reinforces robustness against obstruction and code transformations. Crucially, the interpretability module translates model attention into actionable insights at the source-code level so that human auditors can handily review and act on automated vulnerability alerts. This trade-off between accuracy, localization, robustness and interpretability fills the gap between high-performance AI models such as the BD model and usable auditing workflows in real-world blockchain applications.

Possible directions for future research include the improvement of adversarial robustness through certified defenses and adversarial trained models to defend against more complex evasion techniques. By developing and deploying models alongside CI/CD systems, combined with on-chain monitoring, we can create continuous learning pipelines that are able to adapt to new attack patterns. Studying few-shot and zero-shot adaptation on prompt-augmented large language models with meta-learning techniques may enhance the capability of detections for new vulnerabilities with very little labeled data. Furthermore, integrating detection with automated repair and patch synthesis, verified via symbolic execution and test generation, can save developer work. Finally, making benchmarks uniform, encouraging reproducibility, and designing on privacy- and cost-aware deployment strategy - even with gas-optimized recommendations for both on-chain and off-chain mitigation – further improve the applicability and scalability of AI-driven smart contract security solutions.

| ISSN: 2347-8446 | www.ijarcst.org | editor@ijarcst.org |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 8, Issue 1, January-February 2025||

DOI:10.15662/IJARCST.2025.0801004

REFERENCES

[1] Z. Xu, Y. Liu, and L. Chen, "Deep learning-based vulnerability detection in Ethereum smart contracts," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 2, pp. 489–502, 2023.
[2] Z. Liu, P. Qian, X. Wang, Y. Zhuang, L. Qiu and X. Wang, "Combining Graph Neural Networks With Expert Knowledge for Smart Contract Vulnerability Detection," in IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 2, pp. 1296-1310, 1 Feb. 2023, doi: 10.1109/TKDE.2021.3095196.
[3] D. Joshi, S. Patil, S. Chauhan, T. Baware, R. Thakur and S. Naik, "Smart Contract Vulnerability detection using Natural Language Processing," 2023 International Conference on Recent Advances in Science and Engineering Technology (ICRASET), B G NAGARA, India, 2023, pp. 1-6, doi: 10.1109/ICRASET59632.2023.10420108.
[4] Mishra, S. Blockchain and Machine Learning-Based Hybrid IDS to Protect Smart Networks and Preserve Privacy. *Electronics* 2023, *12*, 3524. https://doi.org/10.3390/electronics12163524
[5] T. -T. -H. Le, J. Kim, S. Lee and H. Kim, "Robust Vulnerability Detection in Solidity-Based Ethereum Smart Contracts Using Fine-Tuned Transformer Encoder Models," in IEEE Access, vol. 12, pp. 154700-154717, 2024, doi: 10.1109/ACCESS.2024.3482389.
[6] P. Praitheeshan, L. Pan, J. Yu, J. Liu, and R. Doss, "Security Analysis Methods on Ethereum Smart Contract Vulnerabilities: A Survey," Aug. 2019, doi: https://doi.org/10.48550/arxiv.1908.08605.
[7] P. Weber, K. V. Carl, and O. Hinz, ''Applications of explainable artificial intelligence in finance—A systematic review of finance, information systems, and computer science literature,'' Manage. Rev. Quart., vol. 74, no. 2, pp. 867–907, Jun. 2024.
[8] L. Zhang, J. Wang, W. Wang, Z. Jin, Y. Su, and H. Chen, "Smart contract vulnerability detection combined with multi-objective detection," Computer Networks, vol. 217, p. 109289, Nov. 2022, doi: https://doi.org/10.1016/j.comnet.2022.109289.
[9] J. Sah, S. Padma, R. Yanamandra, and M. Irfan, ''Risk management of future of Defi using artificial intelligence as a tool,'' in AI-Driven Decentralized Finance Future Finance. Hershey, PA, USA: IGI Global, 2024, pp. 252–272
[10] F. Louati, F. B. Ktata, and I. Amous, ''Big-IDS: A decentralized multi agent reinforcement learning approach for distributed intrusion detection in big data networks,'' Cluster Comput., vol. 27, no. 5, pp. 6823–6841, Aug. 2024.
[11] J. Su, H.-N. Dai, L. Wang, Z. Zheng, and X. Luo, "Effectively Generating Vulnerable Transaction Sequences in Smart Contracts with Reinforcement Learning-guided Fuzzing," Oct. 2022, doi: https://doi.org/10.1145/3551349.3560429
[12] Celik, Y., Barbero, I., Hodorog, A. et al. Blockchain for energy efficiency training in the construction industry. Educ Inf Technol 29, 323–349 (2024). https://doi.org/10.1007/s10639-023-12261-y
[13] S. HajiHosseinKhani, A. H. Lashkari, and A. M. Oskui, ''Unveiling vulnerable smart contracts: Toward profiling vulnerable smart contracts using genetic algorithm and generating benchmark dataset,'' Blockchain: Res. Appl., vol. 5, no. 1, Mar. 2024, Art. no. 100171
[14] Bharat Gami, M. Agrawal, D. K. Mishra, Danish Quasim, and Pawan Singh Mehra, "Artificial intelligence-based blockchain solutions for intelligent healthcare: A comprehensive review on privacy preserving techniques," Transactions on Emerging Telecommunications Technologies, vol. 34, no. 9, Jul. 2023, doi: https://doi.org/10.1002/ett.4824.
[15] Z. Gao, "When deep learning meets smart contracts," arXiv (Cornell University), Dec. 2020, doi: https://doi.org/10.1145/3324884.3418918.
[16] Cholevas C, Angeli E, Sereti Z, Mavrikos E, Tsekouras GE. Anomaly Detection in Blockchain Networks Using Unsupervised Learning: A Survey. *Algorithms*. 2024; 17(5):201. https://doi.org/10.3390/a17050201
[17] H. Liu, Y. Fan, L. Feng, and Z. Wei, "Vulnerable smart contract function locating based on Multi-Relational Nested Graph Convolutional Network," Journal of Systems and Software, vol. 204, pp. 111775–111775, Oct. 2023, doi: https://doi.org/10.1016/j.jss.2023.111775.
[18] Peddamukkula, P. K. (2024). The Impact of AI-Driven Automated Underwriting on the Life Insurance Industry. International Journal of Computer Technology and Electronics Communication, 7(5), 9437-9446.
[19] T. Murakami, S. Wu, J.-Z. Zhang, D.-M. Zhang, K. Asano, Y. Otake, and K.-K. Phoon, ''Differential privacy in geotechnical engineering,'' Geodata AI, vol. 1, Sep. 2024, Art. no. 100004.
[20] Q. Umer, J. -W. Li, M. R. Ashraf, R. N. Bashir and H. Ghous, "Ensemble Deep Learning-Based Prediction of Fraudulent Cryptocurrency Transactions," in IEEE Access, vol. 11, pp. 95213-95224, 2023, doi: 10.1109/ACCESS.2023.3310576.
[21] P. Singh, M. Masud, M. S. Hossain, and A. Kaur, "Cross-domain secure data sharing using blockchain for industrial IoT," *Journal of Parallel and Distributed Computing*, vol. 156, pp. 176–184, Oct. 2021, doi: https://doi.org/10.1016/j.jpdc.2021.05.007