Reference	Methodology	Findings	Improvements for Proposed Work
[1] Z. Liu, M. Jiang, S. Zhang, J. Zhang, and Y. Liu, "A Smart Contract Vulnerability Detection Mechanism Based on Deep Learning and Expert Rules," IEEE Access, vol. 11, pp. 1-10, 2023.	Combines deep learning (GNNs) and expert rules for smart contract vulnerability detection, with an EVM-level transaction blocking feature.	Enhances detection accuracy and scalability but increases time overhead due to EVM checks.	Our system will also integrate GNNs but aim for real-time detection with reduced time overhead by using optimized transformer models and efficient feature extraction.
[2] X. Tang, Y. Du, A. Lai, Z. Zhang, and L. Shi, "Lightning Cat: A Deep Learning-based Solution for Smart Contracts Vulnerability Detection," Salus Security, Beijing, 2023.	Uses Optimized-CodeBERT, LSTM, and CNN models for detecting vulnerabilities in smart contracts.	Optimized-CodeBERT shows superior performance in vulnerability detection. Effective preprocessing using CodeBERT improves accuracy.	Our project will incorporate transformers like CodeBERT but also integrate a rule-based layer to reduce false positives and verify detected vulnerabilities.
[3] Zhuang et al., "A Novel Method for Smart Contract Vulnerability Detection Based on Graph Neural Networks," CMC, vol. 79, no. 2, pp. 3024-3040, 2024.	Utilizes GNNs and control flow graphs for detecting smart contract vulnerabilities.	GNNs excel at capturing complex dependencies, outperforming traditional keyword-based detection methods.	Our framework will also leverage GNNs, but we aim to improve real-time performance by integrating a rule-based system for secondary verification.
[4] Wenzhong Yang et al., "GRATDet: Smart Contract Vulnerability Detector Based on Graph Representation and Transformer," CMC, vol. 76, no. 2, pp. 1460-1480, 2023.	Combines graph representation and transformers to capture syntactic and semantic features of smart contract code.	Improved detection accuracy but is resource-intensive, which could hinder scalability.	We will optimize transformer models (e.g., BERT) using pruning techniques to reduce computational overhead while maintaining detection accuracy.
[5] W. Deng et al., "Smart Contract Vulnerability Detection Based on Deep Learning and Multimodal Decision Fusion," Sensors, vol. 23, no. 7246, pp. 1-21, 2023.	Employs deep learning and multimodal decision fusion, integrating control flow graphs, opcodes, and source code for vulnerability detection.	Enhanced detection accuracy but increased complexity due to multimodal approach.	Our hybrid model will focus on integrating control flow graphs with deep learning, aiming to simplify the process by using fewer modalities.
[6] L. Zhang et al., "A Novel Smart Contract Vulnerability Detection Method Based on Information Graph and Ensemble Learning," Sensors, vol. 22, no. 9, pp. 3581, 2022.	Uses ensemble learning with information graphs to detect vulnerabilities in smart contracts.	Outperforms traditional static tools but is complex to implement.	We will incorporate graph-based methods like GNNs but reduce implementation complexity through modular integration with rule-based systems.
[7] J. Huang et al., "Smart Contract Vulnerability Detection Model Based on Multi-Task Learning," Sensors, vol. 22, no. 1829, pp. 1-24, 2022.	Multi-task learning approach using CNNs for vulnerability classification in smart contracts.	Improves detection accuracy but complexity arises from the multi-task nature of the model.	Our work will utilize CNN-LSTM hybrids but simplify model architecture to focus on vulnerability detection without multi-task complexity.

[8] X. Tang et al., "Lightning Cat: A Deep Learning-Based Solution for Detecting Vulnerabilities in Smart Contracts," Scientific Reports, vol. 13, no. 20106, 2023.	Uses CodeBERT, LSTM, and CNN models to detect vulnerabilities, leveraging semantic features from the source code.	High recall and precision using CodeBERT, but with high computational requirements.	We'll adopt transformer-based models like CodeBERT but optimize for real-time capability through quantization and model pruning.
[9] Y. Liu et al., "SC Vulnerability Detection based on SET," Proceedings of the CNCERT 2021, vol. 1506, pp. 193-207, 2022.	Combines symbolic execution with control flow graphs for vulnerability detection in smart contracts.	Achieves high accuracy but symbolic execution is computationally intensive.	We'll use symbolic execution selectively within the rule-based system to ensure it doesn't hinder real-time performance.
[10] R. N. A. Sosu et al., "VdaBSC: A Novel Vulnerability Detection Approach for Blockchain Smart Contract by Dynamic Analysis," IET Software, vol. 1, no. 1, pp. 1-17, 2023.	Utilizes dynamic analysis, runtime batch normalization, and deep learning for detecting vulnerabilities in smart contracts.	Dynamic analysis enhances detection but introduces high computational demands.	We'll focus on static and control-flow analysis to reduce computational demands while maintaining high detection accuracy.
[11] R. N. A. Sosu et al., "A Vulnerability Detection Approach for Automated Smart Contract Using Enhanced Machine Learning Techniques," Research Square, 2022.	Employs machine learning techniques like PSOGSA and TSVM for smart contract vulnerability detection.	Improves detection accuracy but is sensitive to dataset quality and hyperparameters.	Our project will incorporate deep learning but focus on improving generalizability and robustness through synthetic data augmentation.