QuadraCode AI: Smart Contract Vulnerability Detection with Multimodal Representation

Conference Paper · August 2024

DOI: 10.1109/ICCCN61486.2024.10637655

7 authors, including:

Jiblal Upadhya
Middle Tennessee State University
10 PUBLICATIONS **21** CITATIONS
SEE PROFILE

Kritagya Upadhyay
Middle Tennessee State University
30 PUBLICATIONS **273** CITATIONS
SEE PROFILE

Samir Poudel
Middle Tennessee State University
16 PUBLICATIONS **46** CITATIONS
SEE PROFILE

Nahid Hasan
Ahsanullah University of Science and Technology
3 PUBLICATIONS **11** CITATIONS
SEE PROFILE

QuadraCode AI: Smart Contract Vulnerability Detection with Multimodal Representation

Jiblal Upadhya†, Kritagya Upadhyay*, Arpan Sainju*,
Samir Poudel*, Md Nahid Hasan†, Khem Poudel*, Jaishree Ranganathan*
Department of Computer Science*, Computational and Data Science†
Middle Tennessee State University
Murfreesboro, TN, 37132
Email: {ju2i, sp2ai, mh2ay}@mtmail.mtsu.edu,
{kritagya.upadhyay, arpan.sainju, khem.poudel, jaishree.ranganathan}@mtsu.edu

***Abstract*—In recent years, Smart Contracts have gained in popularity, facilitating billions of US Dollars in daily transactions. However, the recent increase in smart contract vulnerabilities threatens to undermine trust in the technology. The study aims to detect and address potential vulnerabilities in smart contracts in blockchain technology through a comprehensive analysis of four principal modalities: Solidity source code, bytecode, opcode, and intermediate representations. This proactive identification of vulnerabilities can contribute to bolstering the security and dependability of blockchain-based systems. In this paper, we propose a novel multimodal Transformer architecture named QuadraCode AI , utilizing these four distinct modalities. Unlike traditional unimodal analysis, multimodal analysis can provide a more holistic understanding of both the semantic and syntactical contexts of smart contracts to identify underlying vulnerabilities. By employing advanced data fusion techniques such as cross-attention and concatenations across 12 different multimodal frameworks, our approach enhances the detection capabilities beyond traditional unimodal approach. Notably, the framework that integrates opcode with bytecode achieves an impressive average F score of 86%, demonstrating the effectiveness of our method.**

***Index Terms*—Smart Contract, Multimodality, QuadraCode AI, Security, Concatenation, Cross Attention, Blockchain, Deep Learning**

I. Introduction and Background

Blockchain is a distributed ledger technology that enables the secure recording and storage of data in a decentralized manner [15]. A smart contract is a self-executing digital ledger that exists across a decentralized blockchain network where the terms of agreement or conditions are written in Solidity programming language [4]. The widespread adoption of smart contracts has revolutionized various industries, facilitating seamless and trustless transactions on blockchain platforms.

However, the surge in smart contract usage has brought to light the pressing need for robust security measures to safeguard against potential vulnerabilities [25]. Fig. 1 depicts notable vulnerabilities within smart contracts that led to major attacks in blockchain technology. The Decentralized Autonomous Organization (DAO) hack of 2016, caused by a Reentrancy Attack in its Ethereum smart contract, allowed attackers to steal millions of dollars worth of Ether [8]. In a Reentrancy Attack, an attacker exploits a vulnerability in the design of a smart contract to repeatedly re-enter a function before the previous invocation completes, potentially manipulating the contract's state in unintended ways [21]. In 2017, the unchecked external call vulnerability in the Parity wallet code allowed an external user to manipulate the smart contract's state in unintended ways, leading to the loss of access to significant amounts of cryptocurrency funds stored in affected wallets [17]. Furthermore, the BZX flash loan exploits in February 2020, which resulted in significant financial losses, was primarily due to a lack of adequate access control mechanisms within the protocol's smart contracts [27]. Specifically, the attackers exploited vulnerabilities related to the authorization mechanisms of the protocol's lending and borrowing functions. Such incidents highlight the devastating consequences that vulnerabilities can have on individuals and organizations within the blockchain ecosystem, increasing distrust in blockchain technology.

Below we present and briefly discuss the major problem motivations of our work:

1) Intricacy and Complexity of Smart Contracts: Once deployed on the blockchain, smart contracts become immutable, prohibiting any alterations or modifications. It is therefore impossible to debug or fix a smart contract once it has been deployed if it has a bug or security flaw. Hence, detecting and addressing security flaws within smart contracts is significant.

Also, the dynamic and decentralized nature of smart con-

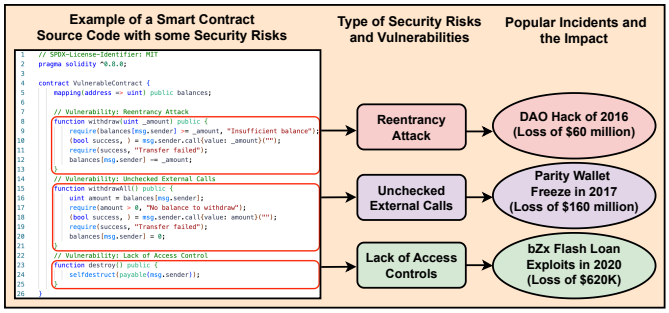

Fig. 1. This figure illustrates how security vulnerabilities can exist in smart contracts written in Solidity and how these vulnerabilities can cause millions of dollars in losses.

tracts creates complex challenges that leave smart contracts more susceptible to exploitation by malicious attackers.

2) Diverse Attack Vectors: Smart contracts are a prime target for malicious actors since they represent a means for high-value transactions to be involved. Attackers exploit plenty of possible loopholes, flaws, and vulnerabilities in the smart contracts such as reentrancy attacks, integer overflows, unexpected contract interactions, and so on in the hope of illegally siphoning funds by manipulating the contract's behavior [25]. In addition, with the evolution of cyber threats and attacks, the range of attacks keeps growing and as a result, we get to see new attacks on a regular basis.

3) Limited Scalability of Manual Audits: When smart contracts are manually audited, it takes more time and effort which often slows down the process of smart contract deployment. Although human intervention at some point in time may be necessary, human auditing may overlook subtle and hidden security risks or even fail to predict emerging threats [25].

4) Adversarial Nature of Smart Contract Ecosystem: The decentralized and pseudonymous nature of the blockchain network ensures that the identities of the parties to transactions remain hidden. Nonetheless, there remains a chance for someone to attempt to establish a connection to the pseudonymous identity through their actions within the blockchain network as attackers are constantly looking for weaknesses in the smart contracts for exploitation. There is still a danger that additional vulnerabilities might emerge as existing vulnerabilities are patched and updated with smart contracts being redeployed on the network. That is why, developers must continuously update their defense systems. However, the primary issue with this strategy is that defense takes a more reactive rather than proactive approach. In current practices, vulnerabilities in smart contracts are primarily identified using various approaches such as manual code review, automated static analysis, symbolic execution, and machine learning-based approaches [1]–[3], [5], [13]. Manual code review relies on human expertise, which can be time-consuming and subjective, potentially overlooking subtle vulnerabilities. Automated static analysis tools like The Mythril, Slither, SmartCheck [33], etc. may produce false positives or false negatives, leading to inaccuracies in vulnerability detection. Symbolic execution requires significant computational resources and may struggle to handle complex smart contract logic effectively. Most existing machine learning-based approaches are based on unimodal analysis. Hence, they may struggle to comprehensively detect all types of vulnerabilities in smart contracts, as they typically focus on specific patterns or signatures of known vulnerabilities.

The focus of our paper is to develop a deep learning-based model based on multimodal analysis to identify vulnerabilities in the smart contract. Multimodal deep learning blends several data sources and metadata to offer a comprehensive strategy that allows the model to identify intricate attack patterns and find hidden correlations that unimodal cannot [16], [18], [24], [26]. Unlike traditional unimodal analysis, multimodal analysis can provide a more holistic understanding of both the semantic and syntactical contexts of smart contracts to identify underlying vulnerabilities [14], [29]. In Fig. 2, we demonstrate the essential role of multimodal representation in identifying smart contracts affected by security risks and vulnerabilities.

We proposed a novel multimodal Transformer network, QuadraCode AI, which integrates multiple sources of data to enhance the analysis and detection of vulnerabilities in smart

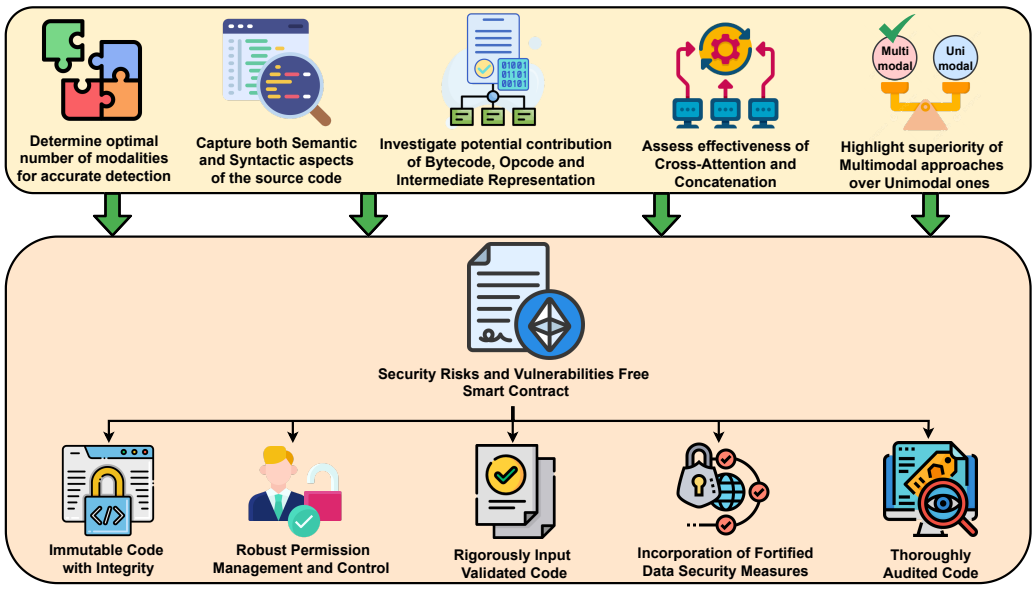

Fig. 2. Illustration to demonstrate the essential role of multimodal representation in identifying smart contracts affected by security risks and vulnerabilities. By leveraging multimodal approaches, we can effectively detect and address these issues, potentially saving millions of dollars. Ultimately, the goal of this work is to ensure that smart contracts are free from security risks and vulnerabilities resulting from rigorous multimodal detection.

contracts. QuadraCode AI leverages four distinct modalities of smart contracts: source code, bytecode, intermediate representation, and opcode. We harness the BERT model to extract features from different combinations of modalities that are used as input in QuadraCode AI.

Furthermore, we investigate the potential significance of each combination of modalities by applying various fusion techniques, aiming to identify the most crucial ones for detecting vulnerabilities in smart contracts using QuadraCode AI. Our investigation primarily focuses on two key data fusion techniques: concatenation and cross-attention.

In summary, we make the following contributions:

- We propose a novel Multimodal Transformer Network, called QuadraCode AI, robust in detecting potential vulnerabilities in smart contracts, thereby enhancing the security and reliability of blockchain-based systems.
- We have contributed with two additional modalities, namely opcode and intermediate representations for a deployed Solidity source code and bytecode in the Slither audited dataset [20] advancing to a total of four modality representations.
- We explored two multimodal data fusion mechanisms for smart contracts, cross-attention, and concatenation, which were able to understand the semantics and syntax of the code with better performance.
- We explored and experimented with all the 12 possible combinations of the multimodal architecture for the proposed system, showing that opcode and bytecode combinations are best in classifying and detecting the vulnerability.
- We have shown that our proposed multimodal approach was able to achieve an avg. F score of 86% outperforming all of the unimodal approaches.

II. Related Work

In this section, we delve into existing research concerning the detection of vulnerabilities in smart contracts, a critical issue within the domain of blockchain security.

Khodadadi et al. [11] explored methodologies for detecting smart contract vulnerabilities by employing diverse input representations, including deep learning techniques such as BiGRU, and leveraging word embedding methodologies. Their investigation revealed that their HyMo model showed strong performance in smart contract vulnerability detection compared to the contemporary models. However, it is noteworthy that the authors restricted their analysis to only two modalities: source code and opcode. Additionally, while their model's accuracy surpassed that of existing models, it still fell marginally below expectations. Likewise, Yuan et al. [31] introduced an entropy embedding technique for addressing vulnerability concerns. However, their study neglected to adequately address issues relating to false positives and negatives, as well as the trade-offs between accuracy and computational efficiency, particularly about the modality gap. Similarly, Tang et al.'s [23] work offers a comprehensive investigation, particularly in the comparison of standalone unimodal approaches. However, their study was limited by the utilization of only three modalities, without incorporating a multimodal framework to ensure a comprehensive evaluation. Yang et al.'s [30] approach notably withheld from integrating multimodal techniques, relying solely on word2vec embeddings, which, compared to BERT, offer limited contextual understanding as they consider only surrounding words in a single direction. Furthermore, their categorization of vulnerabilities was restricted to only six types, ignoring additional popular categories. Duy et al. [9] also adopted a limited approach by utilizing only three modal representations, i.e., source code, opcode, and intermediate representation, without exploring a diverse range of fusion techniques. Their dependence solely on concatenation for data fusion lacks diversity and demands further investigation into the reasoning behind this choice and its comparative usefulness. Finally, we show the comparison of our work's contribution to other existing works in Table I.

III. Methodology

This section introduces a novel multimodal framework called QuadraCode AI.

A. Data Collection

We collect the dataset that contains smart contracts audited by Slither from Hugging Face database [20]. The dataset contains source code and the corresponding bytecode for smart contracts from Etherscan written in Solidity programming language [10]. Furthermore, it also contains a classification

TABLE I
COMPARISON OF OUR WORK'S CONTRIBUTION TO OTHER EXISTING WORKS IN A SIMILAR FIELD

Reference	[12]	[28]	[22]	[34]	[11]	[23]	[32]	[19]	[6]	**Our Work**
Year	2019	2020	2021	2022	2023	2023	2022	2020	2023	2024
Solidity Source code	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Bytecode	X	✓	✓	X	X	X	✓	✓	X	✓
Opcode	✓	✓	X	X	✓	X	✓	✓	✓	✓
Intermediate representation	X	X	X	✓	✓	X	x	X	X	✓
Deep Learning	X	X	X	✓	✓	✓	✓	✓	✓	✓
Multimodality	✓	X	X	X	✓	✓	x	X	✓	✓
Unimodality	X	X	✓	✓	✓	X	X	✓	X	✓
Numerous and Diverse Multimodal Frameworks	X	X	X	X	X	X	X	X	X	✓
Data Fusion Method – (Cross Attention+Concatenation)	X	X	X	X	X	X	X	X	X	✓
Hugging Face (Largest Dataset)	X	X	X	X	X	X	X	X	X	✓

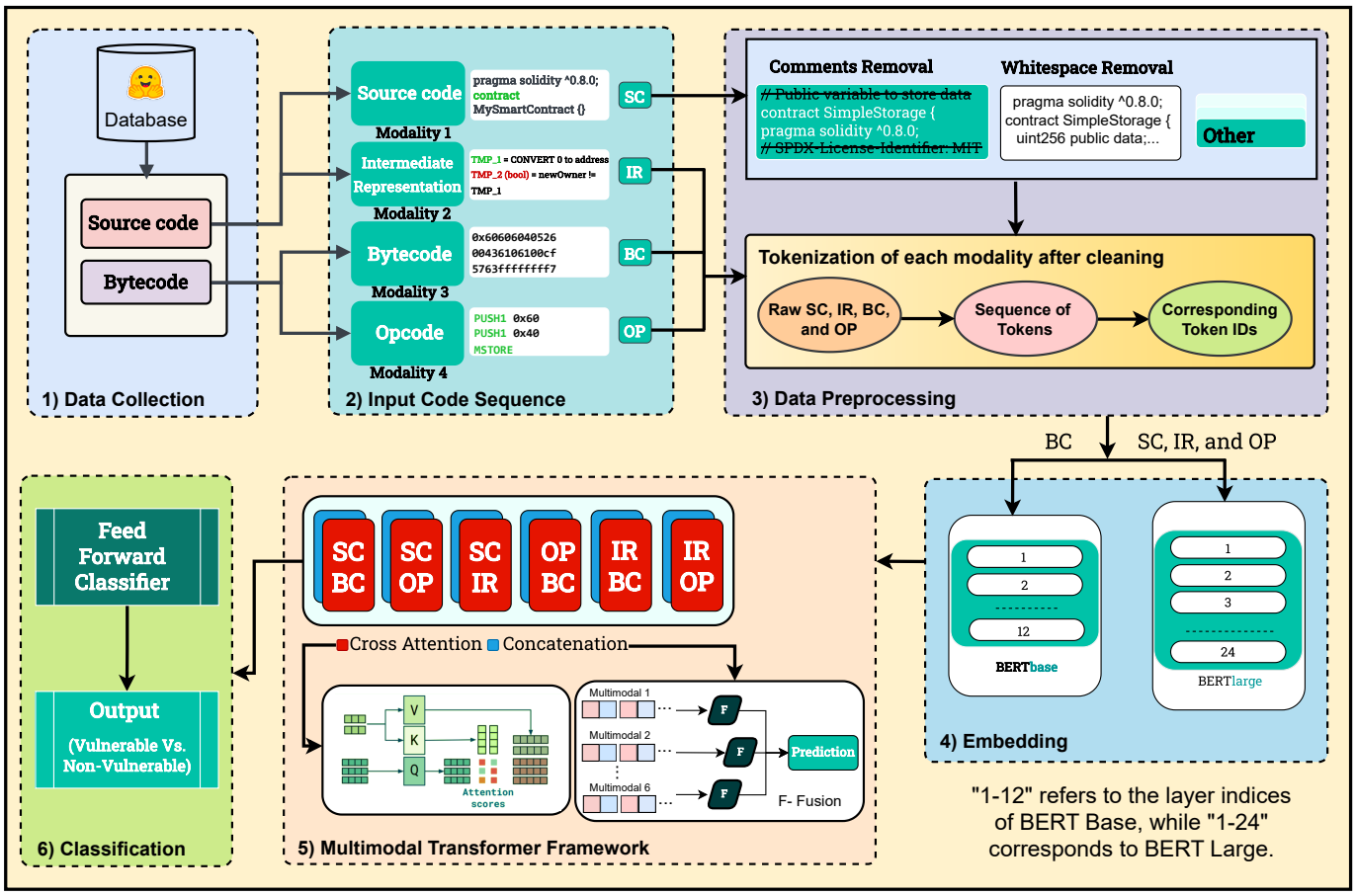

Fig. 3. This figure illustrates a system architecture for a proposed multimodal Transformer to classify security risks and vulnerabilities in deployed smart contracts. It involves extracting smart code and bytecode, processing multiple code representations using a transformer-based framework with BERT embeddings for feature extraction, and concluding with a classifier to predict a smart contract's vulnerability status.

of their vulnerabilities according to the Slither static analysis framework. Each data instance contains the following features: address, source code, and bytecode. In this work, we focus only on binary classification to identify presence or absence of vulnerability on the smart contract.

B. Overall System Architecture

The overall model's architecture is illustrated in Fig. 3. In step 1, we collect the labeled dataset from Hugging Face [20]. The dataset contains 106,000 smart contracts. The dataset contains the following columns:

- **Address**: The Solidity code deployment address.
- **Source code**: The source code of the Solidity code.
- **Bytecode**: The converted bytecode (using Web3.py library) of Solidity code.
- **Slither**: Vulnerability class.

As shown in step 2, we extended the dataset by introducing two additional modalities for each data entry:

- **Opcode**: Assembly code obtained by converting Bytecode using an Opcode-Bytecode converter tool sourced from the Etherscan website.
- **Intermediate representation**: An intermediate representation of the source code obtained by using Slither tool (Solidity and Vyper Static analysis framework).

In step 3, we first preprocess the source code by standard preprocessing techniques such as removing white spaces and comments. Next, we employ tokenization step in which the words in source code, intermediate representation and Opcode are converted into series of tokens. However, we cannot use the standard tokenization method for bytecode because the instuctions are not sperated like words in source code. Hence, for bytecode, tokens are extracted by simply tokenizing the hexadecimal values.

In step 4, we performed embedding transformation to convert tokens into dense vectors within a continuous vector space, leveraging pre-trained models such as BERT (Bidirectional Encoder Representations from Transformers) [7]. BERT is a widely-used language representation model trained on large corpora of text. The key innovation of BERT is its ability to generate contextualized word representations by jointly conditioning on both the left and right context of a word. This bidirectional training approach allows BERT to better capture the semantic meaning and relationships

between words, which is crucial for understanding natural language. BERT comes in different sizes, with 'BERT-base' representing a smaller version and 'BERT-large' a larger and more complex one. The source code, intermediate representation and opcode were embedded using BERT-large model (340M parameters). Whereas, the bytecode was embedded using BERT-base model (110M parameters).

The architecture's core strength lies in its ability to process and merge information from various modalities. Step 5 demonstrates our Multimodal Transformer Framework, in which we implemented two separate data fusion techniques: cross-attention and concatenation. These techniques facilitate the fusion of multimodal data, combining diverse representations from sources such as source code, bytecode, opcode, and intermediate representation. The resulting fused multimodal data is then fed to a feedforward classification model for the final prediction, as shown in Step 6.

C. Multimodal Transfomer Framework

The architecture begins with unimodal feature extraction from source code (SC), bytecode (BC), opcode (OP), and intermediate representation (IR), each processed into uniform vectors. These vectors undergo feature fusion via cross-attention and concatenation, integrating multimodal data into a unified representation. The multimodal feature fusions network being encapsulated through the input features vectors to the classifications task in the process is shown in Fig. 4.

1) Concatenation-Data Fusion Mechanism: Let's denote the output vector of the BERT model for source code as $BERT_{SC}$ and for bytecode as $BERT_{BC}$, with each having dimensions $768 \times L$. The concatenation operation can be represented as:

$$\text{Combined} = \text{Concatenate}(BERT_{SC}, BERT_{BC}) \quad (1)$$

$$\text{Combined} \in \mathbb{R}^{1536 \times L} \quad (2)$$

The sequence of $L \times 1536$ vectors is flattened into a single vector, resulting in a dimension of $L \cdot 1536$. The concatenated vector is then passed through a series of dense blocks (feed forward network) which reduces the dimensions of its input vector., The transformations is represented as:

- Flatten: $\mathbb{R}^{L \times 1536} \rightarrow \mathbb{R}^{L \cdot 1536}$
- Dense Block 1: $\mathbb{R}^{L \cdot 1536} \rightarrow \mathbb{R}^{512}$
- Dense Block 2: $\mathbb{R}^{512} \rightarrow \mathbb{R}^{256}$
- Dense Block 3: $\mathbb{R}^{256} \rightarrow \mathbb{R}^{128}$
- Dense Block 4: $\mathbb{R}^{128} \rightarrow \mathbb{R}^{32}$
- Dense Block 5: $\mathbb{R}^{32} \rightarrow \mathbb{R}^{16}$

The output of the last dense block is passed to a final dense layer with a sigmoid activation function, producing a scalar output, the probability of vulnerability.

2) Cross-Attention Data-Fusion Mechanism: In Cross-Attention Data-Fusion Mechanism, we apply the `CrossAttention` layer to the output embeddings of the BERT models for every pair of modalities. By processing modalities in pairs rather than all at once, the model's computational complexity may be reduced, leading to faster training and inference times. Moreover, grouping modalities in pairs enables the model to learn complex relationships between different modalities. The resulting combination includes SC and BC, SC and IR, SC and OP, BC and IR, BC and OP, and IR and OP.

Given the number of heads (`num_heads`), and the dimensions for keys (`key_dim`) and values (`value_dim`), the dimensionality of the attention outputs for each head will be $L \times$ key_dim. For `num_heads` $= 5$, `key_dim` $= 64$, and `value_dim` $= 64$, each head produces an output of dimension $L \times 64$. These are concatenated to give a tensor of dimensions $L \times 320$.

However, due to the residual connections and layer normalization, the final output dimensionality remains $L \times 768$, consistent with the input embeddings.

The embeddings are flattened, transforming the tensor into a single-dimensional vector, maintaining a size of $L \times 768$. This vector is processed through a series of dense blocks (feed forward network), each reducing the dimensionality as follows:

- Flatten: $\mathbb{R}^{L \times 768} \rightarrow \mathbb{R}^{L \cdot 768}$
- Dense Block 1: $\mathbb{R}^{L \cdot 768} \rightarrow \mathbb{R}^{512}$
- Dense Block 2: $\mathbb{R}^{512} \rightarrow \mathbb{R}^{256}$
- Dense Block 3: $\mathbb{R}^{256} \rightarrow \mathbb{R}^{128}$
- Dense Block 4: $\mathbb{R}^{128} \rightarrow \mathbb{R}^{32}$
- Dense Block 5: $\mathbb{R}^{32} \rightarrow \mathbb{R}^{16}$
- Dense Output: $\mathbb{R}^{16} \rightarrow \mathbb{R}$ with a sigmoid activation function to obtain a probability in the range [0,1].

The final output is a scalar value representing the probability of vulnerability in the smart contract.

D. Unimodal Training

In our current research, we took Big-Multilabel [20] as the datasets, which has the smart contract address, and corresponding to that slither labels are either [4] which means contract is not vulnerable and [0,1,2,3, 5] corresponding to different vulnerable types. For binary classification, we assign the data containing vulnerabilities to the "Vulnerable" class, while those without any vulnerabilities are categorized as "Non-Vulnerable".

TABLE II
CLASS DISTRIBUTION

Dataset	Vulnerable	Non-Vulnerable
Train	6,640	17,668
Test	830	2209
Validation	830	2209

Table II displays the data distribution details for the train, test, and validation datasets, indicating a uniform distribution.

All the modalities were trained on the feed forward network with the same hyperparameters. The loss function was Binary Crossentropy and the Adam Optimizer with 1e-4 learning rate was employed.

Our results, illustrated in Table III, demonstrate various performance metrics, offering a comprehensive comparison

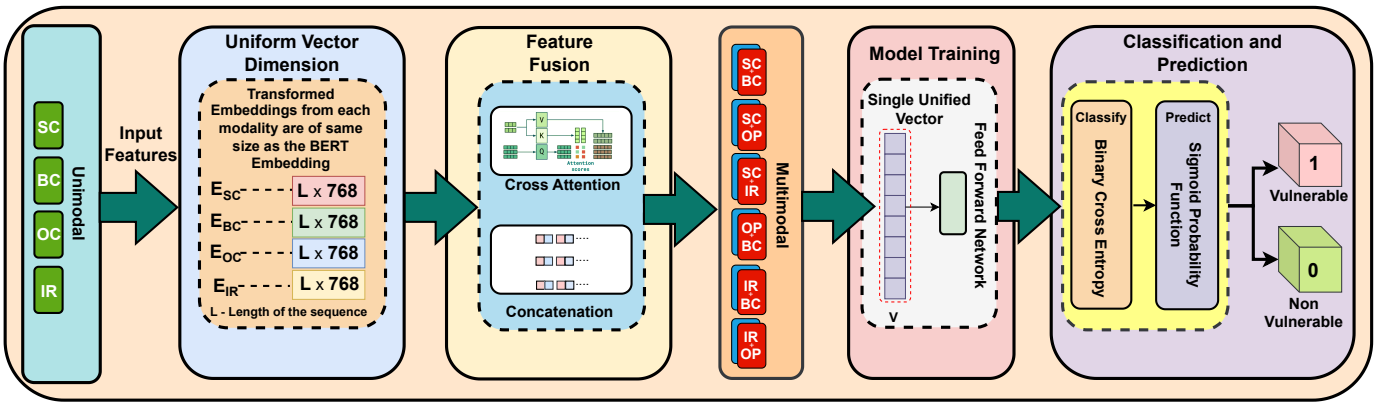

Fig. 4. A Multimodal feature fusion framework for detecting vulnerability in the smart contract. In this process, the input features from each unimodal and the different combinations of the multimodal are converted to the transformed embeddings that pass into the multimodal feature fusion blocks, converting them into a single unified vector for predictions. The steps followed for this are (a) modality features (b) uniform transformed vector dimensions, (c) feature fusion, (d) model training (e) classification (f) prediction.

TABLE III
UNIMODAL PERFORMANCE COMPARISON

Modality	Class	Precision	Recall	F	Avg. F
BC	Non-Vuln.	0.86	0.89	0.88	0.76
	Vuln.	0.68	0.62	0.65	
IR	Non-Vuln.	0.91	0.87	0.89	0.80
	Vuln.	0.68	0.76	0.72	
OP	**Non-Vuln.**	**0.92**	**0.90**	**0.91**	**0.84**
	Vuln.	**0.75**	**0.80**	**0.78**	
SC	Non-Vuln.	0.84	0.91	0.88	0.74
	Vuln.	0.70	0.54	0.61	

of each unimodal representation (SC, BC, IR, and OP) when trained independently.

E. Multimodal Training

In the multimodal setting, we obtain 12 fused dataset after applying the cross attention and concatenation based data fusion technique on every pair of modalities as shown below:

- **SC-BC concat**: Concatenation fusion on SC and BC
- **SC-BC attn**: Cross-attention fusion on SC and BC
- **SC-IR concat**: Concatenation fusion on SC and IR
- **SC-IR attn**: Cross-attention fusion on SC and IR
- **SC-OP concat**: Concatenation fusion on SC and OP
- **SC-OP attn**: Cross-attention fusion on SC and OP
- **BC-IR concat**: Concatenation fusion on BC and IR
- **BC-IR attn**: Cross-attention fusion on BC and IR
- **BC-OP concat**: Concatenation fusion on BC and OP
- **BC-OP attn**: Cross-attention fusion on BC and OP
- **IR-OP concat**: Concatenation fusion on IR and OP
- **IR-OP attn**: Cross-attention fusion on IR and OP

Similar to the unimodal training, All the multi-modalities were trained on the feed forward network with the same hyperparameters. We used Binary Crossentropy loss and Adam Optimizer with a learning rate of 1e-4.

TABLE IV
MULTIMODAL PERFORMANCE COMPARISON

Multimodality	Class	Precision	Recall	F	Avg. F
BC-IR-attn	Non-Vuln.	0.83	0.97	0.89	0.76
	Vuln.	0.84	0.48	0.62	
BC-IR-concat	Non-Vuln.	0.92	0.88	0.90	0.83
	Vuln.	0.72	0.80	0.75	
BC-OP-attn	Non-Vuln.	0.91	0.94	0.92	0.85
	Vuln.	0.83	0.74	0.78	
BC-OP-concat	**Non-Vuln.**	**0.93**	**0.91**	**0.92**	**0.86**
	Vuln.	**0.78**	**0.82**	**0.80**	
IR-OP-attn	Non-Vuln.	0.89	0.91	0.90	0.81
	Vuln.	0.74	0.71	0.72	
IR-OP-concat	**Non-Vuln.**	**0.91**	**0.93**	**0.92**	**0.85**
	Vuln.	**0.81**	**0.75**	**0.78**	
SC-BC-attn	Non-Vuln.	0.74	0.99	0.85	0.50
	Vuln.	0.91	0.08	0.15	
SC-BC-concat	Non-Vuln.	0.78	0.99	0.87	0.63
	Vuln.	0.86	0.25	0.38	
SC-IR-attn	Non-Vuln.	0.84	0.93	0.87	0.76
	Vuln.	0.75	0.54	0.63	
SC-IR-concat	Non-Vuln.	0.83	0.91	0.87	0.72
	Vuln.	0.67	0.50	0.57	
SC-OP-attn	Non-Vuln.	0.88	0.94	0.91	0.82
	Vuln.	0.80	0.67	0.73	
SC-OP-concat	Non-Vuln.	0.86	0.92	0.89	0.78
	Vuln.	0.75	0.60	0.67	

IV. RESULTS AND DISCUSSION

A. Unimodal Analysis

Table III shows the performance results on unimodal data. As shown in the Table III, both of our proposed modality (i.e. opcode and intermediate representation) demonstrated strong performance with average F score of 0.84 and 0.8 respectively. We can also observe that the source code performed worst with the average F score of 0.74. This difference may be attributed to variations in coding styles and variable naming conventions among programmers, even though the source code may perform the same function. This ambiguity is mitigated, if not entirely eliminated, when the source code is translated into lower-level representations. For instance, the

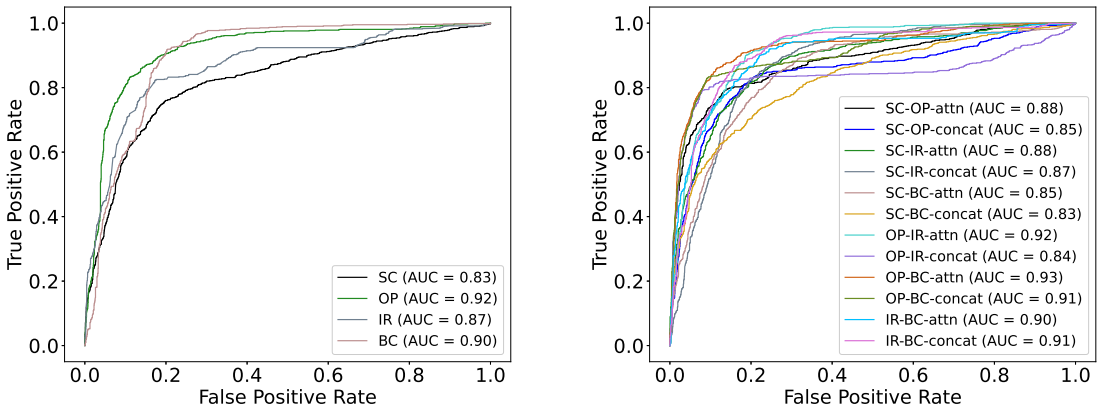

Fig. 5. Receiver Operating Curve for the performance of the smart contract vulnerability detection (a) Unimodal Analysis (b) Multimodal Analysis for all 12 possible combinations of the proposed multimodal Framework. This AUC score for all the multimodality combinations showed that our QuadraCode AI has a strong discriminatory power for vulnerable detection.

opcode representation lacks variable names, instead utilizing a fixed set of register names determined by the compiler. In Fig. 5(a), it is evident that OP exhibits the highest area under the curve (AUC) in the Receiver Operating Characteristic (ROC) curve, while SC displays the lowest AUC. This implies that OP possesses strong discriminatory power, whereas SC exhibits the weakest discriminatory power across various threshold values.

B. Multimodal Analysis

Table IV shows the performance results based on the multimodal setting. We can observe that the multimodal data performed better than the unimodal data. The combination of bytecode with opcode attained the highest performance, achieving average F-scores of 0.86 with concatenation-based fusion and 0.85 with cross-attention-based fusion technique. Additionally, multimodal fusion involving our proposed intermediate representation and opcode, with concatenation-based fusion, yielded comparable average F-scores to the top-performing multimodality. Furthermore, we observe that multimodal data containing source code generally performed less effectively compared to others. The observation is consistent with the Unimodal result, where source code performed the worse. It further strengthens our assumption that source code may not be the optimal choice due to the inherent variability in coding styles among different developers that may lead to ambiguity in the representation. On the other hand, all multimodal combinations involving opcode performed better, with average F-scores ranging from 0.78 to 0.86. This may be attributed to the reduced ambiguity in features associated with multimodalities incorporating lower-level representations, such as the absence of variable names. This reduction in ambiguity could contribute to improved prediction performance. Fig. 5(b) illustrates that that OP-BC with cross-attention, OP-BC with concatenation, OP-IR with cross-attention, IR-BC with cross-attention, and IR-BC with concatenation demonstrate remarkably high AUC values in the ROC curve, ranging from 0.90 to 0.93. This implies that multimodality that includes BC, IR and OP possesses strong discriminatory power across various threshold values.

The multimodal models evaluated in these experiments exhibit a consistent and comprehensive superiority over their unimodal counterparts. The findings distinctly suggest that the fusion of modalities is promising for detecting vulnerabilities in the smart contract vulnerability.

C. Fusion Techniques

In this study, we explored two multimodal fusion techniques, specifically cross-attention based fusion and concatenation fusion, and their impact on the performance of vulnerability detection. We observed that in most cases concatenation based fusion performed better that the cross-attention based fusion technique. One potential reason for this observation could be the nature of the data and the task at hand. Concatenation fusion simply combines the representations from different modalities into a single vector, allowing the model to learn directly from both modalities simultaneously. However, cross-attention-based fusion requires the model to attend to different modalities selectively, which may introduce additional complexity and potentially require longer training time. Ultimately, our research stresses the importance of the selection of fusion methods within multimodal settings, signifying that the right choice of fusion strategies is essential to refine the precision and robustness of smart contract vulnerability detection tools.

V. Conclusion and Future Works

In conclusion, our research presents a significant advancement in enhancing the security and dependability of blockchain-based systems through the development of

QuadraCode AI, a novel multimodal Transformer network tailored for identifying vulnerabilities in smart contracts. By integrating four principal modalities—source code, bytecode, intermediate representation, and opcode—our approach offers a comprehensive understanding of both the semantic and syntactical contexts of smart contracts, enabling robust vulnerability detection. Through extensive experimentation and analysis, we have demonstrated that our proposed model outperforms traditional unimodal-based approaches. Furthermore, our exploration of multimodal fusion techniques, specifically cross-attention and concatenation, contributes to a deeper understanding of the code semantics, leading to improved performance in vulnerability detection.

For our future work, we plan to improve our QuadraCode AI architecture by unifying all four modalities in a single Multimodal fusion framework, which is limited in our present work due to its computational complexities. Similarly, we will investigate multi-class classification for the detection of specific types of vulnerabilities.

REFERENCES

[1] Mauro Argañaraz, Mario Berón, Maria João Pereira, and Pedro Rangel Henriques. Detection of vulnerabilities in smart contracts specifications in ethereum platforms. In *9th Symposium on Languages, Applications and Technologies (SLATE 2020)*, volume 83, pages 1–16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020.

[2] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. A survey of attacks on ethereum smart contracts (sok). In *Principles of Security and Trust: 6th International Conference, POST 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings 6*, pages 164–186. Springer, 2017.

[3] Massimo Bartoletti and Livio Pompianu. An empirical analysis of smart contracts: platforms, applications, and design patterns. In *Financial Cryptography and Data Security: FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers 21*, pages 494–509. Springer, 2017.

[4] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. *white paper*, 3(37):2–1, 2014.

[5] Kevin Delmolino, Mitchell Arnett, Ahmed Kosba, Andrew Miller, and Elaine Shi. Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab. In *Financial Cryptography and Data Security: FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers 20*, pages 79–94. Springer, 2016.

[6] Weichu Deng, Huanchun Wei, Teng Huang, Cong Cao, Yun Peng, and Xuan Hu. Smart contract vulnerability detection based on deep learning and multimodal decision fusion. *Sensors*, 23(16):7246, 2023.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] Vikram Dhillon, David Metcalf, Max Hooper, Vikram Dhillon, David Metcalf, and Max Hooper. The dao hacked. *blockchain enabled applications: Understand the blockchain Ecosystem and How to Make it work for you*, pages 67–78, 2017.

[9] Phan The Duy, Nghi Hoang Khoa, Nguyen Huu Quyen, Le Cong Trinh, Vu Trung Kien, Trinh Minh Hoang, and Van-Hau Pham. Vulnsense: Efficient vulnerability detection in ethereum smart contracts by multimodal learning with graph neural network and language model. *arXiv preprint arXiv:2309.08474*, 2023.

[10] A Etherscan. Etherscan—the ethereum blockchain explorer. *https://etherscan. io/*, 2021.

[11] Mohammad Khodadadi and Jafar Tahmoresnezhad. Hymo: Vulnerability detection in smart contracts using a novel multi-modal hybrid model. *arXiv preprint arXiv:2304.13103*, 2023.

[12] Jian-Wei Liao, Tsung-Ta Tsai, Chia-Kang He, and Chin-Wei Tien. Soliaudit: Smart contract vulnerability assessment based on machine learning and fuzz testing. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, pages 458–465. IEEE, 2019.

[13] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 254–269, 2016.

[14] Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. Attention bottlenecks for multimodal fusion. *Advances in neural information processing systems*, 34:14200–14213, 2021.

[15] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[16] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.

[17] Santiago Palladino. The parity wallet hack explained. *OpenZeppelin blog*, 2017.

[18] Peng Qian, Zhenguang Liu, Qinming He, Butian Huang, Duanzheng Tian, and Xun Wang. Smart contract vulnerability detection technique: A survey. *arXiv preprint arXiv:2209.05872*, 2022.

[19] Peng Qian, Zhenguang Liu, Qinming He, Roger Zimmermann, and Xun Wang. Towards automated reentrancy detection for smart contracts based on sequential models. *IEEE Access*, 8:19685–19695, 2020.

[20] Martina Rossini. Slither audited smart contracts dataset. 2022.

[21] Noama Fatima Samreen and Manar H Alalfi. Reentrancy vulnerability identification in ethereum smart contracts. In *2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, pages 22–29. IEEE, 2020.

[22] Yuhang Sun and Lize Gu. Attention-based machine learning model for smart contract vulnerability detection. In *Journal of physics: conference series*, volume 1820, page 012004. IOP Publishing, 2021.

[23] Xueyan Tang, Yuying Du, Alan Lai, Ze Zhang, and Lingzhi Shi. Deep learning-based solution for smart contract vulnerabilities detection. *Scientific Reports*, 13(1):20106, 2023.

[24] Jiblal Upadhya, Jorge Vargas, Khem Poudel, and Jaishree Ranganathan. Improving the efficiency of multimodal approach for chest x-ray. In *International Conference on Advances in Computing Research*, pages 47–59. Springer, 2024.

[25] Kritagya Upadhyay, Ram Dantu, Yanyan He, Syed Badruddoja, and Abiola Salau. Auditing metaverse requires multimodal deep learning. In *2022 IEEE 4th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA)*, pages 39–46. IEEE, 2022.

[26] Kritagya Upadhyay, Ram Dantu, Yanyan He, Abiola Salau, and Syed Badruddoja. Paradigm shift from paper contracts to smart contracts. In *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 261–268. IEEE, 2021.

[27] Dabao Wang, Siwei Wu, Ziling Lin, Lei Wu, Xingliang Yuan, Yajin Zhou, Haoyu Wang, and Kui Ren. Towards a first step to understand flash loan and its applications in defi ecosystem. In *Proceedings of the Ninth International Workshop on Security in Blockchain and Cloud Computing*, pages 23–28, 2021.

[28] Wei Wang, Jingjing Song, Guangquan Xu, Yidong Li, Hao Wang, and Chunhua Su. Contractward: Automated vulnerability detection models for ethereum smart contracts. *IEEE Transactions on Network Science and Engineering*, 8(2):1133–1144, 2020.

[29] Zhen Yang, Jacky Keung, Xiao Yu, Xiaodong Gu, Zhengyuan Wei, Xiaoxue Ma, and Miao Zhang. A multi-modal transformer-based code summarization approach for smart contracts. In *2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*, pages 1–12. IEEE, 2021.

[30] Zhongju Yang and Weixing Zhu. Improvement and optimization of vulnerability detection methods for ethernet smart contracts. *IEEE Access*, 2023.

[31] Dawei Yuan, Xiaohui Wang, Yao Li, and Tao Zhang. Optimizing smart contract vulnerability detection via multi-modality code and entropy embedding. *Journal of Systems and Software*, 202:111699, 2023.

[32] Lejun Zhang, Jinlong Wang, Weizheng Wang, Zilong Jin, Yansen Su, and Huiling Chen. Smart contract vulnerability detection combined with multi-objective detection. *Computer Networks*, 217:109289, 2022.

[33] William Zhang, Sebastian Banescu, Leonardo Pasos, Steven Stewart, and Vijay Ganesh. Mpro: Combining static and symbolic analysis for scalable testing of smart contract. In *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, pages 456–462. IEEE, 2019.

[34] Qihao Zhou, Kan Zheng, Kuan Zhang, Lu Hou, and Xianbin Wang. Vulnerability analysis of smart contract for blockchain-based iot applications: a machine learning approach. *IEEE Internet of Things Journal*, 9(24):24695–24707, 2022.