基于大语言模型语义增强的多模态智能合约漏洞检测方法研究

康海燕*,樊瑞洋
(北京信息科技大学计算机学院,北京 100192)

摘 要: 近年来,大语言模型(Large Language Models, LLM)的快速发展为智能合约漏洞检测领域带来了新的机遇。为将LLM强大的语言理解能力有效转化为漏洞检测能力,并克服单一模态检测方法的局限性,提出一种基于大语言模型语义增强的多模态智能合约漏洞检测方法(Multimodal Vulnerability detection for smart contracts with LLM enhancement,MVul-L)。该方法融合文本、图结构与视觉三种模态特征,实现了对智能合约语义、结构与上下文信息的深层建模。首先,设计一种用于LLM漏洞检测的任务提示模板,共包含七个关键字段,为LLM提供明确的分析目标、提示策略与输入输出规范,有效减少模型理解偏差。其次,提出一种基于大语言模型和CodeBERT语义增强的文本特征提取方法,LLM通过任务提示模板对合约进行推理解释,生成语义注释的文本输出,将源代码与注释共同输入CodeBERT中,获得合约的文本特征表示。最后,引入图注意力神经网络与卷积神经网络分别对合约结构信息与视觉特征进行建模,并采用基于Transformer的多模态特征融合机制,实现多模态特征间的深度融合。在公开数据集上的实验结果表明,MVul-L较现有方法整体性能更优,在可重入漏洞、时间戳依赖和整数溢出漏洞检测任务中,F1值提升3.51%~9.40%,验证了该方法的有效性。

关键词: 大语言模型;智能合约;漏洞检测;多模态融合;语义增强
基金项目: 国家社会科学基金(No.21BTQ079)
中图分类号: TP309 **文献标识码:** A **文章编号:** 0372-2112(2026)02-0723-11
电子学报 URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20251000

Multimodal Smart Contract Vulnerability Detection Enhanced by Large Language Model Semantics

KANG Haiyan*, FAN Ruiyang
(*College of Computer Science*, *Beijing Information Science and Technology University*, *Beijing* 100192, *China*)

Abstract: The rapid advancement of large language models (LLMs) has created new opportunities for smart contract vulnerability detection. To exploit LLMs' semantic understanding while overcoming single-modal limitations, a multimodal smart contract vulnerability detection method (MVul-L) is proposed. The proposed method integrates textual, structural, and visual modalities for comprehensive modeling of smart contracts. First, a task prompting template for LLM-based vulnerability detection is designed. It contains seven key fields that clearly define analytical objectives, prompting strategies, and input-output specifications, reducing model understanding bias. Second, a semantic-enhanced textual feature extraction method based on LLM reasoning and CodeBERT encoding is developed. The LLM interprets contract logic through the task template and generates semantically annotated textual outputs. Both source code and annotations are fed into CodeBERT to obtain enriched textual representations. Finally, a graph attention network (GAT) and a convolutional neural network (CNN) are employed to model structural and visual features, respectively. A Transformer-based multimodal fusion mechanism is further adopted to achieve deep cross-modal integration. Experimental results on public datasets demonstrate that the overall performance of MVul-L surpasses existing methods. In reentrancy, timestamp dependency, and integer overflow vulnerability detection tasks, the F1 score is improved by 3.51%~9.40%, confirming the effectiveness of the proposed method.

Keywords: large language model; smart contract; vulnerability detection; multimodal fusion; semantic enhancement
Foundation Item(s): National Social Science Foundation of China (No.21BTQ079)

收稿日期:2025-12-30;录用日期:2026-02-06;责任编委:廖清
*通信作者:康海燕

0 引言

随着区块链技术[1-2]的快速发展,智能合约[3-4]作为一种自动执行合同的技术,已经广泛应用于金融、供应链管理、保险等多个领域。然而,智能合约的安全性问题仍然是制约其大规模应用的关键因素之一[5-7]。据统计,近年来发生超过16起智能合约攻击,累计损失超3.8亿美元[8]。因此,如何高效、准确地检测智能合约中的安全漏洞,成为了当前区块链领域的重要挑战[9]。

传统检测方法[10-14]主要依赖于静态分析、符号执行和模糊测试等技术,这些方法在合约的执行路径和数据流分析上具有一定优势。然而,这些方法通常对复杂的、动态的漏洞表现不佳。因此,传统方法在漏洞发现的全面性和高效性方面存在一定的局限性[15]。最近深度学习技术[16]的崛起为智能合约漏洞检测提供了新的解决方案。深度学习方法能够通过学习合约的语义特征,发现潜在漏洞[17]。LLM(Large Language Models)的快速发展,极大地推动了自然语言处理领域的进步[18]。由于智能合约本身包含丰富的语义信息,LLM的优势为智能合约漏洞检测提供了新的思路。一方面,尽管LLM在代码理解方面展现出强大的能力,但在精准检测漏洞方面仍存在一定的局限性。若仅依靠LLM单独进行漏洞检测并模拟专业审计程序,其F1值通常低于45%[19]。另一方面,单一模态的局限性使得模型难以全面捕捉智能合约中的多层次信息[20]。智能合约不仅由文本构成,图结构和视觉特征同样蕴含着重要的漏洞信息[21-22]。然而,现有方法未能有效融合这些结构化和视觉信息,限制了模型在复杂漏洞识别中的表现能力。

本文主要贡献总结如下:

(1)提出一种基于大语言模型语义增强的多模态智能合约漏洞检测方法(Multimodal Vulnerability detection for smart contracts with LLM enhancement,MVul-L)。该方法充分利用大语言模型对代码文本特征进行语义增强,并联合引入图结构特征与视觉模态信息。通过Transformer充分实现多模态特征间的深度融合,从而有效激发LLM的潜在能力,显著提升漏洞检测的准确性与鲁棒性。

(2)设计一种用于LLM漏洞检测的任务提示模板,通过结构化的方式为大语言模型提供明确的输入输出规范,确保模型能够进行有效推理,引导LLM精确生成与漏洞相关的注释和推理。

(3)提出一种基于大语言模型和CodeBERT语义增强的文本特征提取方法。根据智能合约源代码及目标漏洞类型生成任务提示模板,对代码逻辑结构与潜在漏洞风险进行系统化推理,生成语义注释的文本输出。将源代码与LLM生成的注释共同输入CodeBERT中,获得文本特征表示。

1 相关工作

1.1 传统检测方法

智能合约漏洞检测的传统方法主要依赖于静态分析、符号执行和模糊测试等技术。静态分析工具,如Slither[10]和Smartcheck[11],通过对合约源代码的扫描,识别潜在的安全漏洞。Oyente[12]是一种符号执行工具,通过模拟合约的执行过程,分析所有可能的路径,能够发现一些静态分析无法捕捉的漏洞。Mythril[13]则结合了符号执行和模糊测试,深入分析智能合约的控制流。而sFuzz[14]是基于模糊测试的工具,通过自动生成随机输入进行压力测试,能够发现一些难以通过静态分析和符号执行识别的漏洞。

1.2 基于深度学习的方法

基于深度学习的智能合约漏洞检测方法逐渐成为研究的热点。Rechecker[23]是一种基于深度学习的文本分析方法。DR-GCN[24]、TMP[24]和DA-GNN[25]针对合约的图结构进行建模。DR-GCN采用图卷积网络对合约图进行建模;TMP开发了一种结合专家模式和图结构信息的混合模型;DA-GNN则引入了双向图神经网络。这些方法进一步提高了识别能力,但依然依赖于单一模态[26]。VulnSense[21]和TMF-Net[22]则采用多模态融合方法,后者进一步引入Transformer架构,将语义、结构和视觉信息融合,增强了检测能力。

1.3 基于LLM的方法

近年来,LLM的迅速发展为智能合约安全研究带来了新的机遇,ChatGPT[27]等LLM的不断涌现,展现了其在自然语言处理和代码分析方面的强大能力。

Sun等人[28]提出的GPTscan方法,利用LLM的语义理解能力,弥补了传统规则驱动方法的不足。Yu等人[29]则通过将代码翻译为自然语言,并结合预训练模型的文本理解能力来提取漏洞特征。Xia等人[30]则专注于设计定制化的提示模板。Yuan等人[31]利用混合专家模块对LLM进行调优。Jie等人[19]提出的Agent4Vul框架,通过LLM代理系统融合文本和图结构模态,但特征提取与融合方式仍有进一步优化的空间且难以应用在小规模场景中。尽管LLM在智能合约漏洞检测领域展现了独特的技术优势,但其潜力尚未完全挖掘[32]。本节通过整合现有研究,探讨如何充分发挥LLM在代码理解方面的优势,并结合图结构、文本以及视觉模态的特征信息,实现对智能合约漏洞的高效与精准检测。

2　方法设计

总体架构图如图1所示,该方法主要分为四个模块:文本特征提取模块、图特征提取模块、视觉特征提取模块和特征融合模块。其中,文本特征提取模块制造标签信息和提示策略,并添加到任务提示模板中,LLM依据任务提示模板输出源代码和注释,然后源代码和注释通过预训练模型分别进行编码,形成两个特征向量,详见2.1节;图特征提取模块将合约图中的信息通过预训练模型编码后,输送到图注意力神经网络中进行训练,得到图特征向量,详见2.2节;视觉特征提取模块引入LLM,然后将危险函数等特征进行颜色映射,最后通过卷积神经网络得到视觉特征向量,详见2.3节;特征融合模块采用基于Transformer的融合方法,将四种特征向量进行深度融合,并输入全连接层中进行漏洞分类,详见2.4节。

2.1　文本特征提取模块

2.1.1　任务提示模板

为了有效减小LLM理解任务的偏差,同时规范LLM的输出,本文为LLM设计了一套清晰有效的结构化任务提示模板。任务提示模板通过JSON-LD[33]格式定义,以结构化方式规定输入、输出和策略。

(1)任务提示模板关键字段

任务提示模板包含以下7个关键字段:基本信息、角色定义、上下文信息、标签信息、策略层、输入部分、输出规范。表1为一个简略的任务提示模板示例。

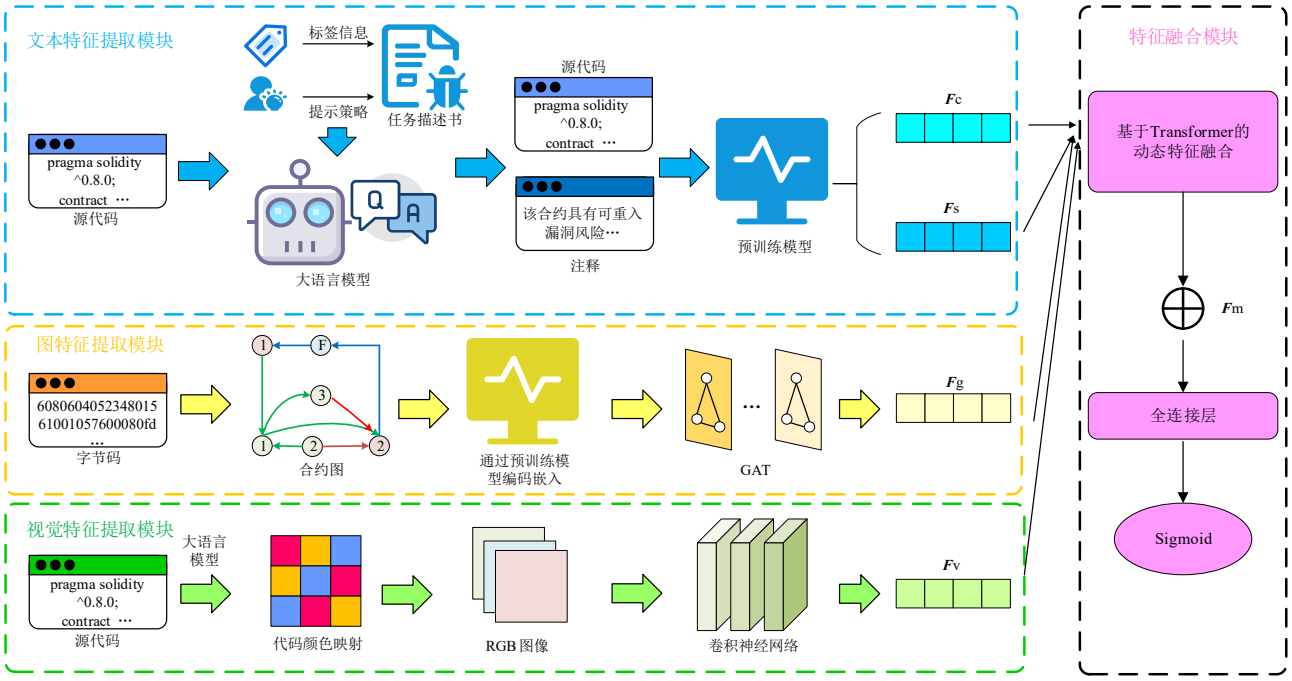


图1　MVul-L的总体架构图

Figure 1　Architecture of MVul-L

表1　任务提示模板描述

Table 1　Description of the task prompt template

字段	字段含义	字段内容示例
@context	JSON-LD中指定词汇或模式定义的命名空间	“@vocab”:“https://example.org/llm-task#”
@type	JSON-LD中定义任务的类型,用于表明当前任务	VulnerabilityDetectionTask
@type	用于标识任务的标题、版本号和语言环境	“title”: “智能合约安全审计任务”, “version”: “1.0”…
角色定义	给模型分配的角色或身份,用于指导其行为方式	“definition”: “你是智能合约安全审计专家,…”
上下文信息	提供区块链平台、编译器版本、依赖库等背景信息	“blockchainPlatform”: “Ethereum”, “compilerVersion”…
标签信息	指定基于规则得到的标签,用于标注漏洞类别或特征	“ruleBasedLabeling”: “ExternalCallBeforeStateUpdate”
策略层	定义模型生成内容时应采用的策略或提示	“selectedStrategy”: “简要描述”, “Prompt”: “请你对合约…”
输入部分	提供任务输入内容,如待分析的源代码	“code”: “pragma solidity ^0.8.0; contract …”
输出规范	要求模型生成内容时必须遵循的输出格式和指令模板	“command”: “请严格按以下结构输出…”

(2)标签信息

标签嵌入[34]是将离散类别标签映射到连续向量空间的技术,该方法在多个领域中取得了显著效果。

本文根据智能合约三类漏洞的触发机制,设计了如表2所示的规则和相应的标签信息。当智能合约符合表中列出的某些规则时,系统将自动识别并将相应的标签添加到任务提示模板中。

(3)策略层

为提升合约代码注释的质量,本文在任务提示模板中引入了策略层。策略层包含四类提示策略:

简要描述策略。该策略以简洁明了的方式为代码生成注释,主要用于快速概览与基本逻辑说明。

详细描述策略。该策略需进行更细粒度的分析,突出变量的使用情况、函数的定义与调用关系。

上下文一致性推理策略。该策略要求模型在生成注释时结合上下文信息,而不仅局限于当前行。例如,对于涉及余额修改与外部调用的代码,模型需要检查该调用在函数整体逻辑中是否存在安全隐患。

基于推理链的漏洞聚焦策略。该策略融合了链式推理范式与漏洞检测需求。在注释生成过程中,模型需通过系统性的分析、推理与验证,逐步演示其对代码逻辑的理解,同时显式标注潜在漏洞。

策略层可根据需求使用不同的组合。例如,在对合约进行初步分析时,可以使用简要描述策略;而在深入的安全审计场景中,则可以同时激活详细描述、上下文一致性推理以及基于推理链的漏洞聚焦策略。

表2 漏洞规则及标签信息

Table 2 Vulnerability rules and label information

漏洞种类	规则	标签
可重入漏洞	存在关键函数	Keyword
	外部调用在状态更新之前	ExternalCallBeforeStateUpdate
	外部调用在循环内	ExternalCallInLoop
	缺少互斥锁/防重入保护	MissingReentrancyGuard
时间戳依赖漏洞	依赖时间戳进行决策	TimestampBasedDecision
	跨区块依赖时间戳	CrossBlockTimestampDependency
	依赖外部数据源和时间戳结合	TimestampExternalDependency
整数溢出漏洞	数字上溢/下溢的边界检查缺失	UncheckedBoundaryCheck
	没有使用安全数学库	MissingSafeMath

2.1.2 文本特征提取

MVul-L的漏洞推理过程如图2所示。首先,根据智能合约源代码匹配对应的漏洞标签,并结合分析目标生成相应的提示策略词,进而构建任务提示模板。其次,LLM在任务提示模板的引导下,基于提示策略和上下文信息进行漏洞推理,生成所需的注释。最后,按照任务提示模板中定义的输出规范,对LLM生成的注释和推理结果进行结构化整合,形成可用于后续语义增强特征提取的标准化输出。

文本特征提取的过程则是首先通过LLM对智能

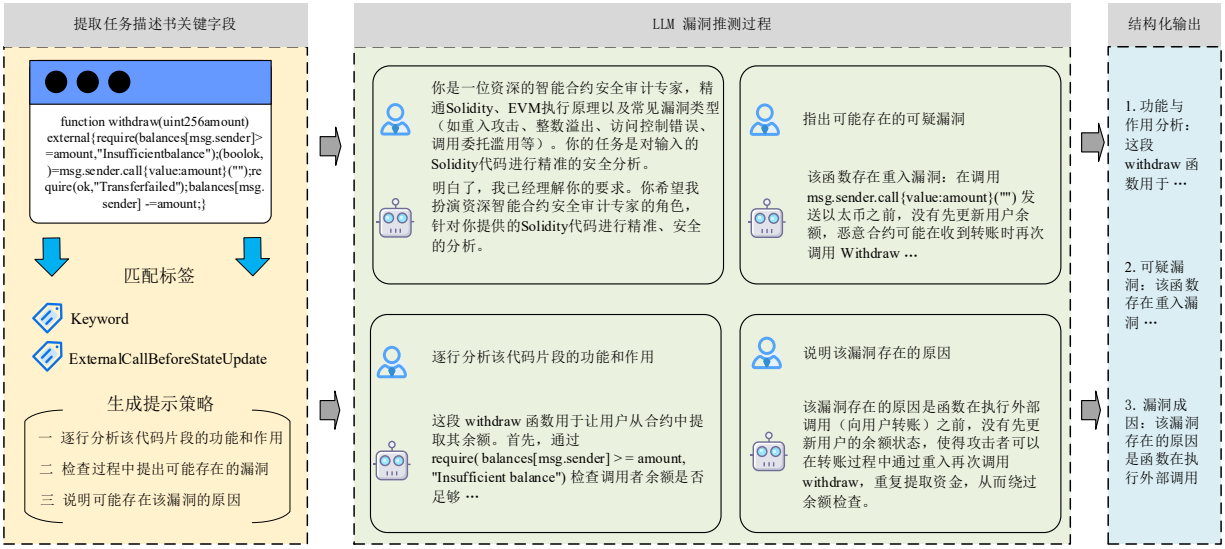


图2 MVul-L漏洞推理过程

Figure 2 Vulnerability reasoning process of MVul-L

合约代码进行分析和注释生成。LLM根据任务提示模板生成结构化的注释,生成的注释具有较强的上下文信息,能够有效补充代码的语义特征。其次,将LLM生成的代码与注释作为输入,送入CodeBERT模型进行编码。CodeBERT是一种专为编程语言设计的预训练语言模型,其基础为Transformer架构,设输入的代码及注释序列为$X=\{x_1,x_2,\cdots,x_n\}$,CodeBERT对其进行嵌入表示$E=\{e_1,e_2,\cdots,e_n\}$,并通过多层自注意力编码器生成上下文感知的特征向量:

$$\boldsymbol{H}=\text{TransformerEncoder}(E) \tag{1}$$

随后,将最后一层隐藏状态或池化后的表示作为代码及注释的高维特征向量$\boldsymbol{v}$:

$$\boldsymbol{v}=\text{Pooling}(\boldsymbol{H}) \tag{2}$$

最终特征向量$\boldsymbol{v}$将作为后续任务的输入,源代码文件到文本特征向量的提取过程如式(3)所示:

$$\boldsymbol{F}_{\text{s}},\boldsymbol{F}_{\text{c}}=\text{TextFeatureExtract}\left(C^{\text{S}}\right) \tag{3}$$

其中:$\boldsymbol{F}_{\text{s}}$、$\boldsymbol{F}_{\text{c}}$分别代表源代码特征和语义注释特征;$C^{\text{S}}$代表源代码。

2.2　图特征提取模块

受到Liu等人[24]的启发,智能合约的源代码可以被建模为合约图,用于捕捉数据和控制流之间的依赖关系。节点代表关键函数,而边则表示它们的执行路径。节点分为三类:主要节点(M)、二级节点(S)和回退节点(F)。主要节点包括对漏洞检测具有重要意义的函数,二级节点代表关键变量,回退节点代表回退函数。边描述了函数间调用路径,边类型包括控制流边、数据流边和回退边,描述了不同类型的依赖关系。

在构造出合约图之后,将每个节点对应的代码片段输入CodeBERT模型进行编码嵌入。设第i个节点对应的代码序列为X_i,其编码向量可表示为

$$\boldsymbol{h}_i^{(0)}=\text{CodeBERT}\left(X_i\right) \tag{4}$$

将由CodeBERT生成的节点嵌入作为输入特征,引入图注意力网络(Graph Attention Network, GAT)。GAT通过注意力机制自适应地分配不同邻居节点的重要性权重,实现对局部语义依赖的动态建模。对于节点i与其邻居节点$j\in N_i$,GAT首先计算注意力系数:

$$e_{ij}=\text{LeakyReLU}\left(\boldsymbol{a}^{\text{T}}\left[\boldsymbol{W}\boldsymbol{h}_i^{(l)}\,/\!/\,\boldsymbol{W}\boldsymbol{h}_j^{(l)}\right]\right) \tag{5}$$

其中:$\boldsymbol{W}$为可学习的线性变换矩阵;$\boldsymbol{a}$为注意力权重向量;$/\!/$表示向量拼接操作。随后,对节点i的所有邻居注意力得分进行归一化:

$$\alpha_{ij}=\frac{\exp\left(e_{ij}\right)}{\sum\limits_{k\in\mathcal{N}_i}\exp\left(e_{ik}\right)} \tag{6}$$

基于上述权重,节点表示通过聚合邻居特征实现更新:

$$\boldsymbol{h}_i^{(l+1)}=\sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}\boldsymbol{W}\boldsymbol{h}_j^{(l)}\right) \tag{7}$$

其中,$\sigma(\cdot)$为非线性激活函数。

字节码文件到图特征向量的提取过程如式(8)所示:

$$\boldsymbol{F}_{\text{g}}=\text{GraphFeatureExtract}\left(B^{\text{S}}\right) \tag{8}$$

其中,$\boldsymbol{F}_{\text{g}}$表示图特征,$B^{\text{S}}$表示字节码。

2.3　视觉特征提取模块

Huang等人[35]提出了基于十六进制字节码映射RGB颜色码生成图像的方法。Wang等人[22]受VSCode中Solidity代码高亮颜色启发,将智能合约代码的不同元素通过颜色映射转换为RGB图像。本文在此基础上,引入LLM深入理解上下文,对存在危险函数等片段赋予特定的颜色映射,生成的图像更加具有语义信息。在特征提取阶段,本文采用卷积神经网络(Convolutional Neural Network, CNN)对语义可视化图像进行多层次特征学习。首先,对输入图像进行标准化处理,以提升模型训练的稳定性并加速收敛。随后,图像经过多层卷积与池化操作,卷积层通过局部感受野捕捉像素间的空间依赖关系,其特征映射可表示为

$$\boldsymbol{X}_{ij}^{(l)}=\sigma\left(\sum_{m,n}\boldsymbol{W}_{mn}^{(l)}\cdot\boldsymbol{X}_{i+m,j+n}^{(l-1)}+b^{(l)}\right) \tag{9}$$

其中:$\boldsymbol{X}_{ij}^{(l)}$表示第$l$层在位置$(i,j)$的输出特征;$\boldsymbol{W}_{mn}^{(l)}$为卷积核参数;$b^{(l)}$为偏置项;$\sigma(\cdot)$为非线性激活函数。池化层进一步通过最大化或平均化操作降低特征维度,公式如下:

$$\boldsymbol{P}_{ij}^{(l)}=\max_{(m,n)\in\Omega}\boldsymbol{X}_{i+m,j+n}^{(l)} \tag{10}$$

其中,Ω表示局部池化窗口。该过程有效去除冗余特征并保留最具代表性的视觉模式。最终,经过卷积与池化层堆叠得到的高维特征映射被展平为视觉特征向量,作为后续多模态漏洞检测模型的输入。源代码文件到视觉特征向量的提取过程如式(11)所示:

$$\boldsymbol{F}_{\text{v}}=\text{VisualFeatureExtract}\left(C^{\text{S}}\right) \tag{11}$$

其中,$\boldsymbol{F}_{\text{v}}$代表视觉特征,$C^{\text{S}}$代表源代码。

2.4　特征融合模块

受Wang等人[22]启发,本文采用基于Transformer[36]架构的特征融合方法,特别是通过引入多头自注意力机制来优化多模态特征的融合过程。通过Transformer的自注意力机制,模型能够捕捉到各个模态之间的相关性,并将这些信息综合到一个特征空间中。此过程如式(12)所示:

$$\boldsymbol{H}=(\boldsymbol{H}_{\mathrm{c}},\boldsymbol{H}_{\mathrm{s}},\boldsymbol{H}_{\mathrm{g}},\boldsymbol{H}_{\mathrm{v}})=\mathrm{Encoder}\left(\boldsymbol{F}_{\mathrm{c}},\boldsymbol{F}_{\mathrm{s}},\boldsymbol{F}_{\mathrm{g}},\boldsymbol{F}_{\mathrm{v}}\right) \tag{12}$$

其中：$\boldsymbol{H}$为编码后的输出，为一个四元组；$\boldsymbol{H}_{\mathrm{c}}$、$\boldsymbol{H}_{\mathrm{s}}$、$\boldsymbol{H}_{\mathrm{g}}$、$\boldsymbol{H}_{\mathrm{v}}$代表不同模态的编码特征。

为增强跨模态融合的稳定性与鲁棒性，在残差分支引入自适应门控，即以系数$\alpha\in(0,1)$控制注意力信息注入强度，该门控在样本噪声较大或模态冲突时自动减小跨模态注入，式(13)~式(15)为计算图模态和其他模态之间相互关系的过程。

$$\mathrm{Attn}(\boldsymbol{G},\boldsymbol{H})=\mathrm{softmax}\left(\frac{\boldsymbol{Q}_{\mathrm{g}}\boldsymbol{K}_{\mathrm{h}}^{\mathrm{T}}}{\sqrt{d_{\mathrm{k}}}}\right)\boldsymbol{V}_{\mathrm{h}} \tag{13}$$

$$\boldsymbol{Z}=\boldsymbol{G}+\alpha\cdot\mathrm{Attn}(\boldsymbol{G},\boldsymbol{H}) \tag{14}$$

$$\boldsymbol{D}_{\mathrm{GH}}=L\left(\boldsymbol{G}+\sigma\boldsymbol{W}^{\mathrm{T}}\left(L(\boldsymbol{Z})\right)+b\right) \tag{15}$$

其中：$\boldsymbol{Q}_{\mathrm{g}}$、$\boldsymbol{K}_{\mathrm{h}}^{\mathrm{T}}$、$\boldsymbol{V}_{\mathrm{h}}$分别表示图模态的查询向量、其他模态的键向量和值向量；L为归一化函数；σ为GELU激活函数；$\boldsymbol{W}^{\mathrm{T}}$、$b$为解码器中用于线性变换的可学习参数。

在解码器对各模态进行自注意力计算后，通过池化操作对融合后的特征进行处理，提取全局信息，得到最终的多模态特征表示。

3 实验与分析

本节对所提出的方法进行了综合评估，主要围绕以下问题展开研究：

RQ1：本方法对比当前最先进的智能合约漏洞检测方法，是否有更好的检测效果？

RQ2：多模态融合策略是否有利于智能合约漏洞检测？

RQ3：CodeBERT相对其他模型的性能如何？

RQ4：不同的LLM对于性能的影响如何？

RQ5：任务提示模板的不同组成要素对于性能的影响如何？

RQ6：基于Transformer的特征融合方式是否起到了增强漏洞检测的效果？

3.1 实验设置

(1)数据集。选取文献[37-38]中公开的智能合约数据集作为实验数据集，重点针对可重入漏洞、时间戳依赖漏洞和整数溢出漏洞进行实验分析。实验选择60%作为训练集、20%作为验证集、20%作为测试集。每个实验重复10次，最后取平均结果。

(2)参数设置。所有实验均在本地环境完成，硬件环境为第13代Intel Core i5-13600KF CPU、32 GB内存和NVIDIA GeForce RTX 4060 GPU。LLM使用GPT-3.5-turbo模型和DeepSeek的V3模型，这两类模型具有先进的自然语言理解能力和推理能力，能有效进行漏洞的推理。训练模型采用交叉熵损失、Adam优化器和学习率调度器对模型进行训练，使用网格搜索法调整了以下超参数：隐藏层大小{64、128、256}，学习率{0.01、0.005、0.001}，批次大小{32、64、128}。最后实验结果根据以下默认超参数的设置进行报告：学习率=0.001，隐藏层=256，epoch=50，batch size=32，dropout=0.1。

(3)评估指标。本文根据准确率(ACCuracy，ACC)、精确率(PREcision，PRE)、召回率(RECall，REC)和F1值来评估方法的性能。

3.2 性能对比实验 (RQ1)

(1)与传统漏洞检测方法的对比。本文选取了五类传统的漏洞检测工具：Smartcheck、Oyente、Mythril、Slither和sFuzz。Smartcheck是一种基于静态分析的检测工具，具体为Oyente是以符号执行为核心的检测工具；Mythril是一款结合符号执行、污点分析和控制流分析的检测工具；Slither是以规则驱动的静态检测工具；sFuzz是一个专门面向智能合约的模糊测试工具。详细的实验结果如表3和图3所示。

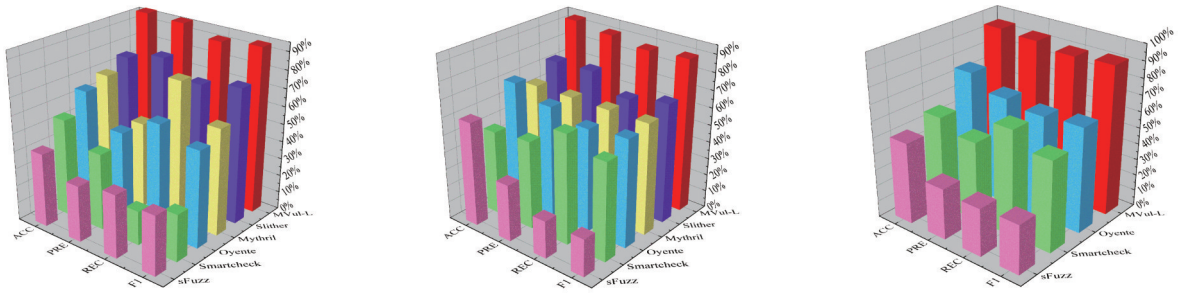
表3 与传统的漏洞检测工具的对比 单位：%

Table 3 Comparison with traditional vulnerability detection tools unit: %

工具	可重入漏洞				时间戳依赖漏洞				整数溢出漏洞			
	ACC	PRE	REC	F1	ACC	PRE	REC	F1	ACC	PRE	REC	F1
Smartcheck	55.10	43.22	19.41	26.73	49.02	51.31	63.84	56.85	56.40	49.11	65.22	56.02
Oyente	66.02	49.30	61.40	54.69	71.61	65.10	60.20	62.55	75.13	66.40	63.33	64.81
Mythril	70.10	48.91	79.30	60.45	64.79	64.85	64.82	64.81	—	—	—	—
Slither	75.91	81.31	72.42	76.59	73.82	74.62	65.21	69.59	—	—	—	—
sFuzz	41.72	31.81	35.64	33.59	60.43	33.15	21.92	21.89	50.21	31.52	29.60	30.52
MVul-L	**97.10**	**96.31**	**91.22**	**93.68**	**93.25**	**90.31**	**87.45**	**88.83**	**94.59**	**93.11**	**90.19**	**91.62**

注：加粗数据为最优结果。

实验结果分析如下。传统工具大多存在检测能力不均衡、精确率与召回率失衡等问题。Slither在可重入漏洞检测中表现较好，受益于其规则驱动的分析方式；Oyente和Mythril有较高的召回率，但精确率相对受限。时间戳依赖漏洞整体检测效果较低，表明该类漏洞对语义理解要求较高。部分工具未支持整数



(a) 可重入漏洞　(b) 时间戳依赖漏洞　(c) 整数溢出漏洞

(a) Reentrancy　(b) Timestamp dependency　(c) Integer overflow

图3　与传统的漏洞检测工具的对比

Figure 3　Comparison with traditional vulnerability detection tools

溢出漏洞检测。相比之下,MVul-L在三类漏洞上均具有显著优于传统工具的检测性能。

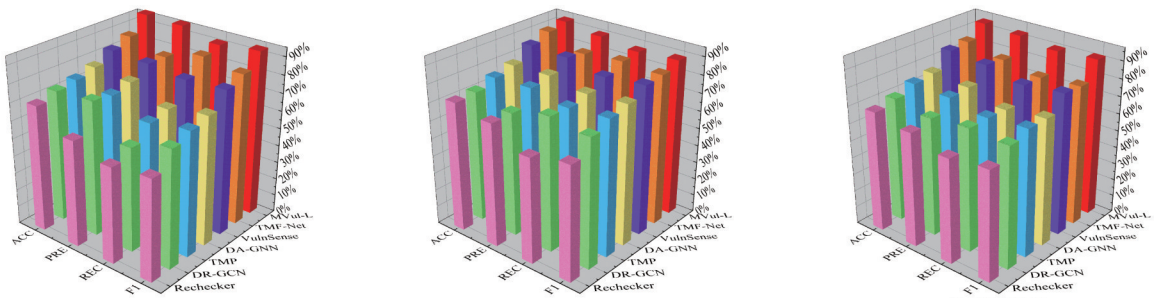
(2)与基于深度学习的方法对比。本文将与五类基于深度学习的方法(Rechecker、DR-GCN、TMP、DA-GNN、VulnSense和TMF-Net)进行对比。Rechecker只对文本序列进行分析;DR-GCN、TMP和DA-GNN分别采用了不同神经网络对合约图进行建模;VulnSense采用多模态的方法结合图信息和文本信息进行检测;TMF-Net通过Transformer将文本特征、图特征和视觉特征进行融合。详细的实验结果如表4和图4所示。

表4　与基于深度学习方法的对比　　单位:%

Table 4　Comparison with deep learning-based methods　　unit: %

方法	可重入漏洞				时间戳依赖漏洞				整数溢出漏洞			
	ACC	PRE	REC	F1	ACC	PRE	REC	F1	ACC	PRE	REC	F1
Rechecker	71.43	60.18	54.27	57.02	73.26	69.57	59.38	64.06	69.84	66.18	60.47	63.19
DR-GCN	74.68	76.24	58.97	66.55	74.17	69.86	75.48	72.58	72.36	68.79	71.18	69.96
TMP	76.42	74.60	66.91	70.43	77.65	78.64	74.98	76.75	76.29	75.45	71.58	73.45
DA-GNN	79.18	77.29	69.48	73.19	80.27	80.97	77.68	79.30	77.86	76.47	70.89	73.58
VulnSense	84.37	83.18	80.69	81.92	87.29	86.48	82.19	84.28	86.57	84.68	79.87	82.22
TMF-Net	88.47	82.67	88.97	85.72	91.17	84.38	86.29	85.32	87.68	83.19	79.48	81.29
MVul-L	**97.10**	**96.31**	**91.22**	**93.68**	**93.25**	**90.31**	**87.45**	**88.83**	**94.59**	**93.11**	**90.19**	**91.62**

注:加粗数据为最优结果。



(a) 可重入漏洞　(b) 时间戳依赖漏洞　(c) 整数溢出漏洞

(a) Reentrancy　(b) Timestamp dependency　(c) Integer overflow

图4　与基于深度学习的方法对比

Figure 4　Comparison with deep learning-based methods

实验结果分析如下。Rechecker的整体检测性能低,原因在于该方法仅基于文本进行建模。而DR-GCN、TMP和DA-GNN引入了图神经网络。其中,DA-GNN通过更精细的图结构建模取得了更稳定的效果。VulnSense利用文本与图结构特征,提升模型的能力。TMF-Net采用Transformer架构进行深度融合,其F1值有明显提升。MVul-L通过引入大语言模型进行语义增强,在F1值等多项指标上均显著优于对比

方法。综上所述，MVul-L 在三类任务中均取得了最优得分，证明了其在智能合约漏洞检测任务中的有效性。

3.3 多模态策略的有效性实验（RQ2）

为了系统验证所提多模态融合策略的有效性，本文设计了一组消融实验，对比以下模型变体，通过对比实验，可以揭示不同模态在漏洞检测中的贡献程度，并验证跨模态特征融合对整体性能提升的作用。

T-only：仅保留文本模态特征，完全忽略图结构与视觉信息。

G-only：仅使用图神经网络对合约结构进行建模，不考虑源代码的语义特征与视觉特征。

V-only：仅利用基于 RGB 映射的视觉特征，舍弃文本与图信息。

实验结果如表 5 所示，分析如下。单模态方法的整体性能均低于 MVul-L，T-only 与 G-only 的表现接近。文本模态经过语义增强后，其 F1 值在可重入和时间戳依赖漏洞的检测中优于视觉模态，甚至接近图模态，这说明 LLM 使得文本模态显著增强。综上所述，融合三类模态的 MVul-L 在任务中均取得最优结果。这表明多模态策略能够有效结合文本语义特征、图结构信息以及视觉模式表达，从而提升检测性能。

表 5 不同的模态的对比实验 单位：%

Table 5 Comparison of different modalities unit: %

模态	可重入漏洞				时间戳依赖漏洞				整数溢出漏洞			
	ACC	PRE	REC	F1	ACC	PRE	REC	F1	ACC	PRE	REC	F1
T-only	62.87	65.94	47.38	55.12	74.92	70.58	59.63	64.64	70.68	61.34	49.82	54.83
G-only	69.84	69.21	50.12	58.14	75.63	68.74	77.08	72.65	77.19	76.58	59.21	66.82
V-only	66.91	66.24	47.59	55.36	72.48	73.11	65.02	68.83	74.06	56.71	75.19	64.68
MVul-L	**97.10**	**96.31**	**91.22**	**93.68**	**93.25**	**90.31**	**87.45**	**88.83**	**94.59**	**93.11**	**90.19**	**91.62**

注：加粗数据为最优结果。

3.4 CodeBERT 与其他模型的对比实验（RQ3）

为了进一步验证 CodeBERT 在智能合约漏洞检测任务中的有效性，本文对其与其他主流模型进行了对比实验。模型选取 GraphCodeBERT[39] 和 CodeT5[40] 作为对比对象，GraphCodeBERT 属于结构增强的预训练模型，CodeT5 属于 seq2seq 代码预训练模型。二者代表当前代码预训练中两类常见架构路线，因此本文选择这两个模型进行对比。通过在相同的数据集和实验设置下对比，评估不同模型在本文方法上的性能差异。

实验结果如表 6 所示，分析如下。GraphCodeBERT 与 CodeT5 的整体性能仅较 CodeBERT 略有提升，因为本任务模型是对源代码和 LLM 语义注释进行文本编码，而不是进行结构建模或生成式推理，使得 GraphCodeBERT 的结构优势被显著弱化，CodeT5 的生成能力也无法发挥有效作用。综上所述，CodeBERT 作为轻量、高效的编码器已能满足本框架的需求，引入更复杂的模型不会带来实质性性能提升。

表 6 不同模型的对比实验 单位：%

Table 6 Comparative experiments on different models unit: %

模态	可重入漏洞				时间戳依赖漏洞				整数溢出漏洞			
	ACC	PRE	REC	F1	ACC	PRE	REC	F1	ACC	PRE	REC	F1
GraphCodeBERT	95.01	84.62	95.03	89.53	94.88	93.12	93.54	93.33	96.21	88.34	95.02	91.53
CodeT5	95.23	85.41	95.28	90.07	95.02	93.87	93.92	93.89	96.18	88.95	95.14	91.91
MVul-L (CodeBERT)	**97.10**	**96.31**	**91.22**	**93.68**	**93.25**	**90.31**	**87.45**	**88.83**	**94.59**	**93.11**	**90.19**	**91.62**

注：加粗数据为最优结果。

3.5 不同 LLM 的影响性实验（RQ4）

为了进一步探究不同 LLM 对 MVul-L 的影响，本文设计了不同 LLM 的对比实验。实验中，分别采用 GPT-3.5-turbo 与 DeepSeek-V3 实现漏洞检测任务，保持其他模块与参数配置不变。

实验结果如表 7 所示，分析如下。整体上，MVul-L（DeepSeek-V3）的检测效果优于 GPT-3.5-turbo，原因在于 DeepSeek-V3 对中文任务提示模板与提示策略的理解更为准确，能够生成更贴近语境的语义注释，从而增强文本模态的表达能力。综上所述，在以中文为核心的任务下，不同的 LLM 呈现出的实验结果不同。

3.6 任务提示模板的影响性实验（RQ5）

为探究任务提示模板在 MVul-L 中的作用，本文进一步设计了消融实验，对比以下三种模型变体：

No-Desc：完全移除结构化任务提示模板，直接将源代码输入 LLM，并布置任务。

表7　不同LLM的对比实验　　单位:%

Table 7　Comparative experiments on different LLMs　　unit: %

LLM	可重入漏洞				时间戳依赖漏洞				整数溢出漏洞			
	ACC	PRE	REC	F1	ACC	PRE	REC	F1	ACC	PRE	REC	F1
GPT-3.5-turbo	86.01	84.72	83.18	83.94	87.96	85.63	81.92	83.74	87.62	80.11	84.96	82.47
MVul-L (DeepSeek-V3)	**97.10**	**96.31**	**91.22**	**93.68**	**93.25**	**90.31**	**87.45**	**88.83**	**94.59**	**93.11**	**90.19**	**91.62**

注:加粗数据为最优结果。

No-Tag:去除标签信息,即保留任务提示模板的主体结构,但不提供规则匹配生成的漏洞标签。

No-Prompt:去除提示策略,即不对LLM进行策略引导。

实验结果如表8所示,分析如下。No-Desc的性能下降最为明显,在整数溢出漏洞检测中,F1值仅为79.60%。No-Tag的性能略低于完整模型,但整体仍保持较高水平。No-Prompt的性能同样有所下降,可重入和时间戳依赖漏洞的F1值均低于MVul-L,说明提示策略有效影响LLM的性能。综上所述,任务提示模板发挥了关键作用,其中提示策略在提升LLM性能方面贡献更大,而标签信息贡献较小。

表8　任务提示模板对MVul-L的影响　　单位:%

Table 8　Impact of the task prompt template on MVul-L　　unit: %

模型变体	可重入漏洞				时间戳依赖漏洞				整数溢出漏洞			
	ACC	PRE	REC	F1	ACC	PRE	REC	F1	ACC	PRE	REC	F1
No-Desc	86.42	81.37	85.92	83.58	88.15	82.46	84.71	83.57	85.63	80.28	78.94	79.60
No-Tag	90.87	84.69	90.14	87.33	91.24	88.73	89.52	89.12	93.16	85.91	91.27	88.51
No-Prompt	88.03	82.91	87.48	85.13	90.46	83.94	86.12	85.01	88.92	83.67	81.05	82.34
MVul-L	**97.10**	**96.31**	**91.22**	**93.68**	**93.25**	**90.31**	**87.45**	**88.83**	**94.59**	**93.11**	**90.19**	**91.62**

注:加粗数据为最优结果。

3.7　特征融合方式的有效性实验(RQ6)

本小节与主流的多模态融合技术对比。所有模型均保持一致的参数配置,确保公平性与可重复性。对比两类方法:直接特征拼接(Direct Feature Concatenation, FDC),将模态的特征向量直接拼接成一个长向量;特征加权拼接(Feature Weighted Concatenation, FWC),在拼接前对每种模态分配特定权重。

实验结果如表9所示,分析如下。MVul-L均显著优于FDC和FWC两种方法。FDC方法仅依赖简单拼接,将图特征和文本特征直接串联;FWC在拼接前引入了权重分配,但本质上依旧缺乏对跨模态依赖关系的有效利用。相比之下,MVul-L所采用的基于Transformer的融合方式表现最优。综上所述,MVul-L所采用的基于Transformer的融合方式可以有效融合多模态信息,提升强模型对智能合约漏洞的检测能力。

表9　不同融合方式的对比实验　　单位:%

Table 9　Comparative experiments on different fusion methods　　unit: %

融合方法	可重入漏洞				时间戳依赖漏洞				整数溢出漏洞			
	ACC	PRE	REC	F1	ACC	PRE	REC	F1	ACC	PRE	REC	F1
FDC	80.94	72.18	73.64	72.90	70.83	76.41	71.26	73.75	70.12	63.47	57.92	60.56
FWC	84.36	86.02	72.95	78.93	83.91	76.38	77.64	77.01	83.48	74.62	73.18	73.89
MVul-L	**97.10**	**96.31**	**91.22**	**93.68**	**93.25**	**90.31**	**87.45**	**88.83**	**94.59**	**93.11**	**90.19**	**91.62**

注:加粗数据为最优结果。

4　结论

本文为有效利用大语言模型的代码理解能力和逻辑思考能力,并针对现有方法存在单一模态的局限性问题,提出了基于大语言模型语义增强的多模态智能合约漏洞检测方法(MVul-L),该方法设计了一种任务提示模板,融入了标签信息和提示策略,以结构化LLM的输入输出。根据设计的任务提示模板,提出了一种基于大语言模型和CodeBERT语义增强的文本特征提取方法,生成语义增强的文本特征向量。通过基于Transformer的特征融合方式将文本信息、图信息和视觉信息进行深度融合,显著提高漏洞检测性能。实验结果表明,MVul-L在可重入漏洞、时间戳依赖漏洞和整数溢出漏洞的检测中的性能优于现有先进方法。

参考文献

[1] Liu Yang, He Jinlong, Li Xiangyang, et al. An overview of blockchain smart contract execution mechanism[J]. Journal of Industrial Information Integration, 2024, 41: 100674.

[2] Kang Haiyan, Wu Bing, Cao Yiran. Research on efficient data processing method for fog computing based on blockchain and federated learning[J]. Neurocomputing, 2025, 626: 129529.

[3] Wu Guangfu, Wang Haiping, Lai Xin, et al. A comprehensive survey of smart contract security: State of the art and research directions[J]. Journal of Network and Computer Applications, 2024, 226: 103882.

[4] Vidal F R, Ivaki N, Laranjeiro N. Vulnerability detection techniques for smart contracts: A systematic literature review[J]. Journal of Systems and Software, 2024, 217: 112160.

[5] Zheng Zibin, Xie Shaoan, Dai Hongning, et al. An overview on smart contracts: Challenges, advances and platforms[J]. Future Generation Computer Systems, 2020, 105: 475-491.

[6] Crisostomo J, Bacao F, Lobo V. Machine learning methods for detecting smart contracts vulnerabilities within Ethereum blockchain – A review[J]. Expert Systems with Applications, 2025, 268: 126353.

[7] Ali Khan Z, Namin A S. A survey of vulnerability detection techniques by smart contract tools[J]. IEEE Access, 2024, 12: 70870-70910.

[8] DefiLlama. Defillama hacks[EB/OL]. [2025-12-30]. https://defillama.com/hacks.

[9] Chu Hanting, Zhang Pengcheng, Dong Hai, et al. A survey on smart contract vulnerabilities: Data sources, detection and repair[J]. Information and Software Technology, 2023, 159: 107221.

[10] Feist J, Grieco G, Groce A. Slither: A static analysis framework for smart contracts[C]//2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain. Piscataway: IEEE, 2019: 8-15.

[11] Tikhomirov S, Voskresenskaya E, Ivanitskiy I, et al. SmartCheck: Static analysis of ethereum smart contracts[C]//Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain. New York: ACM, 2018: 9-16.

[12] Luu L, Chu D H, Olickel H, et al. Making smart contracts smarter[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, 2016: 254-269.

[13] Durieux T, Ferreira J F, Abreu R, et al. Empirical review of automated analysis tools on 47, 587 Ethereum smart contracts[C]//Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. New York: ACM, 2020: 530-541.

[14] Nguyen T D, Pham L H, Sun Jun, et al. sFuzz: An efficient adaptive fuzzer for solidity smart contracts[C]//Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. New York: ACM, 2020: 778-788.

[15] Kushwaha S S, Joshi S, Singh D, et al. Systematic review of security vulnerabilities in ethereum blockchain smart contract[J]. IEEE Access, 2022, 10: 6605-6621.

[16] 康海燕, 王骁识. 基于数据特征相关性和自适应差分隐私的深度学习方法研究[J]. 电子学报, 2024, 52(6): 1963-1976.
Kang Haiyan, Wang Xiaoshi. Research on the deep learning method based on data feature relevance and adaptive differential privacy[J]. Acta Electronica Sinica, 2024, 52(6): 1963-1976. (in Chinese)

[17] Al Ghanmi H, Ahmadjee S, Bahsoon R,et al. Exploring human-centric dimensions in blockchain smart contracts[J]. ACM Computing Surveys, 2026, 58(7): 1-43.

[18] 张亚洲, 刘祈蒙, 戎璐, 等. 语音大模型: 架构、训练与挑战分析[J]. 电子学报, 2025, 53(9): 3454-3472.
Zhang Yazhou, Liu Qimeng, Rong Lu, et al. Speech large language models: Architecture, training and challenges analysis[J]. Acta Electronica Sinica, 2025, 53(9): 3454-3472. (in Chinese)

[19] Jie Wanqing, Qiu Wangjie, Yang Haofu, et al. Agent4Vul: Multimodal LLM agents for smart contract vulnerability detection[J]. Science China Information Sciences, 2025, 68(6): 160101.

[20] 黄辰, 刘会杰, 张龑, 等. 基于自适应噪声和方面图关联学习增强多模态方面级情感分析[J]. 电子学报, 2025, 53(9): 3397-3409.
Huang Chen, Liu Huijie, Zhang Yan, et al. Enhancing multimodal aspect-based sentiment analysis with adaptive noise and aspect graph association learning[J]. Acta Electronica Sinica, 2025, 53(9): 3397-3409. (in Chinese)

[21] Duy P T, Khoa N H, Quyen N H, et al. Vulnsense: Efficient vulnerability detection in ethereum smart contracts by multimodal learning with graph neural network and language model[J]. International Journal of Information Security, 2024, 24(1): 48.

[22] Wang Tengfei, Zhao Xiangfu, Zhang Jiarui. TMF-Net: Multimodal smart contract vulnerability detection based on multiscale transformer fusion[J]. Information Fusion, 2025, 122: 103189.

[23] Qian Peng, Liu Zhenguang, He Qinming, et al. Towards automated reentrancy detection for smart contracts based on sequential models[J]. IEEE Access, 2020, 8: 19685-19695.
[24] Zhuang Yuan, Liu Zhenguang, Qian Peng, et al. Smart contract vulnerability detection using graph neural network[C]// Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. New York: ACM, 2020: 3283-3290.
[25] Zhen Zixain, Zhao Xiangfu, Zhang Jinkai, et al. DA-GNN: A smart contract vulnerability detection method based on Dual Attention Graph Neural Network[J]. Computer Networks, 2024, 242: 110238.
[26] Zhang Yifan, Kang Haiyan, Wang Qiang. MMFDetect: Webshell evasion detect method based on multimodal feature fusion[J]. Electronics, 2025, 14(3): 416.
[27] Kalla D, Smith N, Samaah F, et al. Study and analysis of chat GPT and its impact on different fields of study[J]. International journal of innovative science and research technology, 2023, 8(3): 3639117.
[28] Sun Yuqiang, Wu Daoyuan, Xue Yue, et al. GPTScan: Detecting logic vulnerabilities in smart contracts by combining GPT with program analysis[C]//Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. New York: ACM, 2024: 1-13.
[29] Yu Lei, Lu Junyi, Liu Xianglong, et al. PSCVFinder: A prompt-tuning based framework for smart contract vulnerability detection[C]//2023 IEEE 34th International Symposium on Software Reliability Engineering. Piscataway: IEEE, 2023: 556-567.
[30] Xia Shihao, Shao Shuai, He Mengting, et al. AuditGPT: Auditing smart contracts with ChatGPT[PP/OL]. V1. arXiv (2024-04-05)[2025-12-30]. https://doi.org/10.48550/arXiv.2404.04306.
[31] Yuan Hang, Yu Lei, Huang Zhirong, et al. MOS: Towards effective smart contract vulnerability detection through mixture-of-experts tuning of large language models[PP/OL]. V1. arXiv (2025-04-16)[2025-12-30]. https://doi.org/10.48550/arXiv.2504.12234.
[32] Vukić N, Petrović V B, Dragan D, et al. LLM-based tooling for smart contract auditing[C]//2025 12th International Conference on Electrical, Electronic and Computing Engineering. Piscataway: IEEE, 2025: 11114151.
[33] Wang Jinbo, Aryani A, Wyborn L, et al. Providing research graph data in JSON-LD using schema.org[C]//Proceedings of the 26th International Conference on World Wide Web Companion. New York: ACM, 2017: 1213-1218.
[34] Li Jandong, Fu Jia. A multi-label text classification model combining label graph embedding and transformer decoding[C]//2023 5th International Academic Exchange Conference on Science and Technology Innovation. Piscataway: IEEE, 2023: 375-379.
[35] Huang Hsien-De T T, Kao H Y. R2-d2: Color-inspired convolutional neural network (cnn)-based android malware detections[C]//2018 IEEE International Conference on Big Data. Piscataway: IEEE, 2018: 2633-2642.
[36] Kang Hai, Wang Yunhe, Chen Hanting, et al. A survey on vision transformer[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023, 45(1): 87-110.
[37] Qian Peng, Liu Zhenguang, Yin Yifan, et al. Cross-modality mutual learning for enhancing smart contract vulnerability detection on bytecode[C]//Proceedings of the ACM Web Conference 2023. New York: ACM, 2023: 2220-2229.
[38] Liu Zhengguang, Qian Peng, Yang Jiaxu, et al. Rethinking smart contract fuzzing: Fuzzing with invocation ordering and important branch revisiting[PP/OL]. V2. arXiv (2023-01-12) [2025-12-30]. https://doi.org/10.48550/arXiv.2301.03943.
[39] Guo Daya, Ren Shuo, Lu Shuai, et al. GraphCodeBERT: Pre-training code representations with data flow[PP/OL]. V4. arXiv (2021-09-13) [2025-12-30]. https://doi.org/10.48550/arXiv.2009.08366.
[40] Wang Yue, Wang Weishi, Joty S, et al. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation[C]//Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. Stroudsburg: ACL, 2021: 8696-8708.

作者简介

康海燕　男,1971年生,河北石家庄人。博士,北京信息科技大学计算机学院教授。主要研究方向为网络安全与隐私计算等。
E-mail: kanghaiyan@126.com

樊瑞洋　男,2000年生,山东东营人。北京信息科技大学网络空间安全专业硕士研究生。主要研究方向为网络安全与隐私保护等。
E-mail: ruiyangfan@126.com