计算机科学
COMPUTER SCIENCE

基于异构合约图多维度特征深度融合的漏洞检测方法

周涛, 杜永萍, 谢润锋, 韩红桂

引用本文

周涛, 杜永萍, 谢润锋, 韩红桂. 基于异构合约图多维度特征深度融合的漏洞检测方法[J]. 计算机科学, 2025, 52(9): 368-375.

ZHOU Tao, DU Yongping, XIE Runfeng, HAN Honggui. Vulnerability Detection Method Based on Deep Fusion of Multi-dimensional Features from Heterogeneous Contract Graphs [J]. Computer Science, 2025, 52(9): 368-375.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.241000007

基于异构合约图多维度特征深度融合的漏洞检测方法

周 涛 杜永萍 谢润锋 韩红桂
北京工业大学计算机学院 北京 100124
(zhoutao_work@163.com)

摘 要 智能合约是在区块链上自动执行的代码,具有不可逆性且与金融交易密切相关,故其安全问题至关重要。然而,当前智能合约漏洞检测技术仍面临特征提取效率低、检测精度低以及过度依赖专家规则等问题。对此,提出一种基于异构合约图多维度特征深度融合的漏洞检测方法。首先,针对智能合约数据集的代码进行去噪处理,采用代码函数交换等数据增强方法扩充数据集,进而将其表示为异构合约图。其次,结合图嵌入技术以及代码预训练技术高效获取智能合约图以及对应操作码中节点的高维度语义表示。最后,设计双层异构图注意力网络深度融合在两种维度下学习到的节点特征,以实现高效的漏洞检测。针对不同类型漏洞的实验结果表明,所提方法整体表现较对比方法均有所提升,F1 指标平均值高于 77.72%,在拒绝服务漏洞类型的检测上表现最佳,F1 值最高达到 84.88%,较传统的深度学习方法和图拓扑检测方法分别提升了 10.62%和 22.34%。所提方法不仅提高了检测的效率和准确性,而且通过学习节点特征减少了对专家规则的依赖,为智能合约的安全性提供了更为可靠的保障。

关键词:智能合约;预训练模型;图嵌入;图注意力网络;漏洞检测;区块链
中图分类号 TP309

Vulnerability Detection Method Based on Deep Fusion of Multi-dimensional Features from Heterogeneous Contract Graphs

ZHOU Tao,DU Yongping,XIE Runfeng and HAN Honggui
College of Computer Science,Beijing University of Technology,Beijing 100124,China

Abstract Smart contracts are pieces of code that execute automatically on the blockchain,and the safety problem is critical due to their irreversibility and close links to financial transactions. However,the current smart contract vulnerability detection technology still faces problems such as low feature extraction efficiency,low detection accuracy,and over-reliance on expert rules. In order to solve these problems,this paper proposes a vulnerability detection method based on multi-dimensional feature deep fusion of heterogeneous contract graph. Firstly,the code of smart contract data is denoised,and the data set is expanded by data enhancement method of code function exchange,and represented as heterogeneous contract graph. Secondly,the high-dimensional semantic representation of nodes in the smart contract graph is efficiently obtained by combining graph embedding technology and code pre-training technology. Finally,the dual heterogeneous graph attention network is designed to deeply integrate the node features learned in two dimensions to achieve more accurate vulnerability detection. The experimental results for different types of vulnerabilities show that the overall performance of the proposed method has been improved,and the average F1 index is higher than 77.72%. In the case of denial of service vulnerability detection,the F1 value is up to 84.88%,which is significantly improved by 10.62% and 22.34% compared with the traditional deep learning method and the graph topology detection method respectively. The proposed method not only improves the detection efficiency and accuracy,but also reduces the dependence on expert rules by learning node characteristics,providing a more reliable guarantee for the security of smart contracts.

Keywords Smart contract,Pre-trained model,Graph embedding,Graph attention network,Vulnerability detection,Blockchain

1 引言

智能合约是运行在区块链上的代码,基于计算机通信协议进行信息验证或执行。由于区块链技术的防篡改性,智能合约一旦部署就无法修改[1]。为了减少智能合约漏洞可能导致的重大损失,许多研究者致力于研究智能合约的安全性问题并提出了漏洞检测方法。目前,主流的传统智能合约检测工具主要采用静态分析和符号执行等特定分析方法,该技术

到稿日期:2024-10-08 返修日期:2025-02-25
基金项目:国家重点研发计划(2022YFB3305802);国家自然科学基金(92267107)
This work was supported by the National Key Research and Development Program of China(2022YFB3305802) and National Natural Science Foundation of China(92267107).
通信作者:韩红桂(rechardhan@bjut.edu.cn)

很大程度上依赖于一系列预设的规则或模式来识别潜在的安全问题，而规则或模式通常基于专家知识、安全最佳实践以及常见的漏洞类型[2]。随着智能合约复杂性的增加，基于规则的传统检测方法开始显现出局限性。深度学习是一种新兴的漏洞检测方法，但在特征提取效率和检测精度方面仍面临挑战。本文面向区块链平台以太坊智能合约，提出一种基于异构合约图多维度特征深度融合的漏洞检测方法。本文的主要贡献如下：

1）提出了智能合约转换为异构合约图的方法，将合约操作码转换为基于抽象类型的节点表示，能进一步从中提取智能合约代码的深层逻辑信息。

2）采用图嵌入方法和基于代码的预训练语言模型，分别对图结构信息与操作码文本序列两种维度的高级语义信息进行特征表示，有效理解合约代码的行为逻辑及潜在风险。

3）实现了对 6 种不同类型漏洞的检测，在拒绝服务类型漏洞检测上 F1 值达到 84.88%，较传统的深度学习方法和图拓扑检测方法分别提升了 10.62%和 22.34%。

2　相关工作

2.1　图神经网络方法

随着智能合约不断发展，其结构愈加复杂，基于漏洞定义的专家规则已无法适应智能合约漏洞检测的需求，将智能合约代码转为图表示的深度学习方法逐步应用于该任务中。Nguyen 等[3]提出了 MANDO-HGT 网络结构，引入 Transformer[4]以提高对异构合约图的信息提取能力。Cai 等[5]通过构建抽象语法树和程序依赖图的方式并基于程序切片技术进行规范化，使用混合图神经网络模型识别合约潜在漏洞。Luo 等[6]提出了 SCV 框架，利用图注意力网络来检测智能合约中的漏洞，利用专家知识来辅助自动捕捉与漏洞相关的信息。Pasqua 等[7]开发了工具 EtherSolve，通过解析 EVM 字节码中的跳转地址构建精确的控制流图对合约漏洞进行检测。Zhen 等[8]将智能合约控制流图中的操作码序列转换为特征矩阵，并利用注意力机制更新节点嵌入，再通过自注意力机制融合信息以获取图特征，从而对合约漏洞进行分类检测。

本文采用的图嵌入神经网络是一种将图的节点和边特征映射到低维向量空间的方法。由于图结构的复杂性，其主要挑战是如何保留网络拓扑和节点信息，以在深度学习等任务中获得更好的性能，通过图嵌入方法学习复杂图中节点的结构信息，从而提供更丰富的输入表示。

2.2　基于预训练模型的代码漏洞检测

预训练语言模型具有广泛应用，例如 ELMo[9]，GPT[10]，BERT[11]等模型，通过在大型无监督文本语料库上进行预训练并在下游任务进行微调，展现出优异性能。GraphCodeBERT[12]是一种预训练编程语言模型，与传统的抽象语法树模型不同，其更多考虑了代码的结构和数据流图。CodeBERT[13]使用 Transformer 架构进行预训练，采用自注意力机制捕捉代码上下文和依赖关系。此外，该方法通过预测掩蔽代码片段中被掩蔽部分的内容等方式，学习代码的语义表示和上下文信息。本文采用预训练模型 CodeBERT 来提取合约操作码的特征表示。将操作码序列与图结构中的节点对应，通过学习合约代码的语义表示和上下文信息，将操作码片段映射为固定长度的向量表示，进一步获取代码在语义层面的特征。

3　模型结构

面向智能合约操作码，本文提出基于异构合约图多维度特征深度融合的漏洞检测方法（Multi-dimensional feature Fusion vulnerability detection method based on Heterogeneous contract Graph Attention network，MFHGA）。该方法由 3 个主要模块构成。1）合约图创建模块：通过去噪与数据增强等操作对智能合约源码进行预处理，并将其表示为智能合约图。2）异构节点多维度特征生成模块：设计图嵌入神经网络与预训练模型获取每个节点的图结构特征和文本特征表示，丰富节点的特征表达。3）基于多维度特征融合的双层异构图注意力网络漏洞检测模块：采用融合策略将节点的图结构特征与语义特征进行深度融合，利用注意力机制学习元路径以及节点特征中所蕴含的智能合约代码的深层特征，最后使用全连接层神经网络进行漏洞检测与分类。整体结构如图 1 所示。

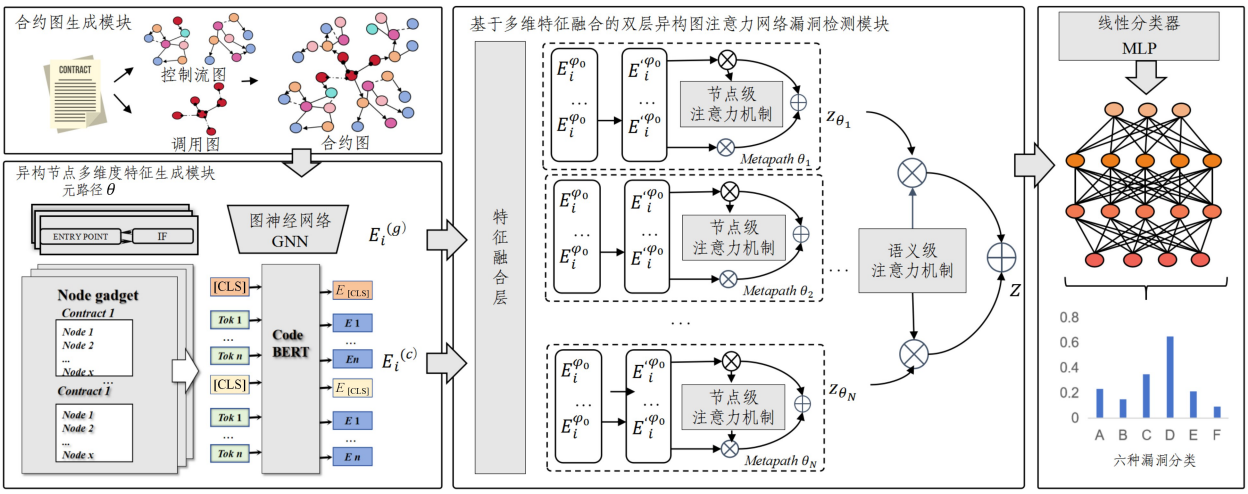

图 1　基于异构合约图多维度特征深度融合的漏洞检测方法

Fig. 1　Multi-dimensional feature fusion vulnerability detection method based on heterogeneous contract graph attention network

3.1 合约图创建模块

3.1.1 智能合约代码表示

对智能合约源代码的预处理包括两个主要步骤：去噪和数据增强。

1)去噪：通过两级处理来解决智能合约中的噪声问题。在代码级别，删除在语义上与代码无关的语句，例如无效注释、代码之间的空行、未使用的变量和函数等；在合同级别上，从代码中删除不影响合同的基本操作的代码，例如未使用的外部合约引用、合约代码版本号、无意义的条件判断等。

2)数据增强：为了增加特定类型样本的数量，结合交换和插入操作来扩充数据集，从而创建一个由无漏洞合约中的函数组成的函数池，然后随机选择这些函数插入其他合约中，形成新的数据样本。此外，通过交换合约内部的函数产生具有不同操作码顺序的序列，进一步丰富了漏洞数据样本的多样性。

在代码的图表示中，将智能合约中一段连续执行的操作码序列视为一个基本块。每个节点代表操作码中的一个基本块，每条边则代表控制流之间的转移关系。每个节点对应一种节点类型，主要包括 FUNCTIONENTRY，EXPRESSION，VARIABLE，RETURN，IF，END IF，LOOP 和 ENDLOOP 这 8 种类型。节点 v 表示为 $\{v_{\text{block}}, T_v\}$，其中 v_{block} 描述节点代码块的操作码信息，节点类型标签 T_v 由漏洞检测工具生成。

3.1.2 异构合约图生成

合约图由合约代码对应的基本控制流图(Control Flow Graph，CFG)和调用图(Call Graph，CG)融合得到。使用漏洞检测分析工具 Slither[14] 遍历和分析每个以太坊智能合约的操作码，生成两种图结构。其中，控制流图 CFG 用于表示智能合约函数中的代码跳转控制关系，每个节点代表智能合约操作码序列中的一个基本块，每条边用于描述代码执行的顺序或分支结构，捕捉程序的逻辑和控制流信息。在异构控制流图 CFG 中，其控制流节点和控制边具有不同的类型，表示它们的语义信息和关系，表达式为：

$$G_{\text{CFG}} = \{V', E', T_v', T_e'\} \tag{1}$$

其中，V' 代表 CFG 图中的顶点合集，E' 代表其边合集，每个节点 v_i 属于顶点集 V'，每个节点都对应节点类型 T_v'。对于控制流图 CFG，控制边有 3 种类型：Next Jump 边、True Jump 边和 False Jump 边，分别表示代码的顺序执行、判断语句的真假跳转。

调用图 CG 用于表示来自同一个或不同智能合约的函数之间的调用关系。智能合约对应生成调用图，表示为：

$$G_{\text{call}} = \{V'', E'', T_v'', T_e''\} \tag{2}$$

其中，V'' 代表调用图中的节点集，E'' 代表其边集。调用图只关注两种调用方式，即合约内部函数调用(Inner Call)和从当前合约到其他合约函数的外部调用(Outer Call)。

为了学习丰富的图结构信息，将智能合约生成的两种图结构 CFG 和 CG 进一步融合为全局的智能合约图(Fusion Contract Graph，FCG)，表示为：

$$G_{\text{fusion}} = \{V, E, T_v, T_e\} \tag{3}$$

在进行融合操作时，以调用图 CG 中的每个函数的入口节点 FUNCTIONENTRY 作为融合桥梁，将对应函数的控制流图 CFG 附加到函数入口节点上，遍历所有 CFG 与 CG 建立节点映射关系，并将其标签及相关信息建立映射字典。通过节点映射字典将控制流图和调用图的信息进行融合，合约图 FCG 的结构如图 2 所示。

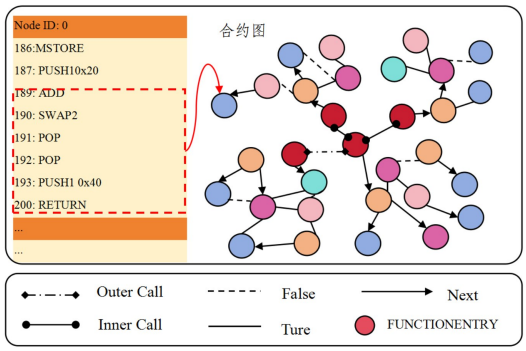

图 2 基于智能合约操作码基本块的合约图构建

Fig. 2 Construction of contract graph based on basic blocks of smart contract opcode

3.2 异构图节点多维度特征生成模块

3.2.1 合约图元路径

元路径(Meta-path)由图中不同类型节点和边组成，用于捕捉节点之间的关系。节点间的元路径描述为：

$$\theta_{ij} : v_i \rightarrow v_{i+1} \rightarrow \cdots \rightarrow v_j \tag{4}$$

对合约图中起始节点类型和结束节点类型相同的元路径进行提取，采用长度为 2 的元路径进行评估，将长度为 N 的元路径分解为 $N-1$ 个长度为 2 的元路径，通过节点之间的邻接关系连接来提取多个元路径。

3.2.2 多维度语义节点特征提取

为了获得智能合约节点的图结构信息，采用拓扑图嵌入神经网络来学习输入图的拓扑结构。通过图嵌入网络卷积层从节点 v_i 的局部邻域中聚合信息，并利用节点 v_i 的邻居节点 v_j 的特征及两者的连接关系来更新并获得节点 v_i 的特征表示。经过与权重矩阵参数 $\boldsymbol{W}_g$ 和偏置参数 b_g 进行运算操作，得到经维度转换后的节点特征 $E_i^{(g)}$，如式(5)所示：

$$E_i^{(g)} = \boldsymbol{W}_g[GNN(G_{f\text{usion}}\{v_i, e, T_v, T_e\})] + b_g \tag{5}$$

为了获得合约图节点代码文本语义层面的信息，利用预训练语言模型为每个节点生成对应操作码文本的嵌入表示。将节点集合所对应的操作码序列块 v_{block} 作为输入，使用 CodeBERT 模型训练得到代码文本级别的节点语义特征 $E_i^{(c)}$。将节点 v_i 对应的操作码片段 v_{block} 作为编码器的输入，学习到的节点语义特征通过线性变化得到与节点图特征一致的维度，如式(6)所示：

$$E_i^{(c)} = \boldsymbol{W}_c CodeBERT(v_i) + b_c \tag{6}$$

3.3 基于多维特征融合的双层异构图注意力网络漏洞检测模块

为了获取合约代码更深层次的信息，对上述基于图结构的节点特征 $E_i^{(g)}$ 和基于操作码语义信息的节点特征 $E_i^{(c)}$ 进行深度融合。设计特征融合策略，为两种不同特征分配不同比例的权重，如式(7)和式(8)所示：

$$\alpha_i = sigmoid(E_i^{(c)} L_1 + E_i^{(g)} L_2) \tag{7}$$

$$E_i = \alpha_i E_i^{(c)} + (1-\alpha_i) E_i^{(g)} \tag{8}$$

其中，α_i为融合权重，L_1和L_2为两个权重参数，节点v_i的融合特征表示为E_i。

由于合约图的异构性，不同类型的节点具有不同的特征表示，节点之间难以进行关联和比较。本文对每一种节点类型φ_i设计了对应的转换矩阵$\boldsymbol{M}_{\varphi i}$，将不同类型的节点特征映射到相同的特征空间，如式(9)所示：

$$E_i' = \boldsymbol{M}_{\varphi i} \cdot E_i \tag{9}$$

其中，E_i与E_i'分别表示节点v_i的原始特征和映射后特征。

对于相邻的节点v_i和v_j，两者之间的元路径连接表示为θ_{ij}。利用节点注意力衡量在元路径中一个节点对其邻接点的重要程度，权重$E_{i,j}^{\theta}$表示元路径θ_{ij}下节点v_j对节点v_i的重要性。

$$E_{i,j}^{\theta} = Att_{\text{node}}(E_i', E_j', \theta_{ij}) \tag{10}$$

其中，Att_{node}表示节点级注意力的深度神经网络。由于合约异构图具有不对称性，因此学习到的节点间注意力权重也不对称。本文采用节点级注意力获取到节点v_i在元路径θ_{ij}下的邻居节点v_j的注意力权重，邻居节点v_j为所有基于元路径θ_{ij}的节点集合$v_j \in N_{v_i}^{\theta}$，并使用 softmax 函数归一化得到每个特征向量的权重系数$\alpha_{i,j}^{\theta}$，最终获得节点v_i基于元路径$\theta_{i,j}$的特征z_i^{θ}，如式(11)和式(12)所示：

$$\alpha_{i,j}^{\theta} = \text{softmax}(E_{i,j}^{\theta}) = \frac{\exp(\sigma(att_{\theta}^{T} \cdot [E_i'][E_j']))}{\sum_{v_j \in N_{v_i}^{\theta}} \exp(\sigma(att_{\theta}^{T} \cdot [E_i'][E_j']))} \tag{11}$$

$$z_i^{\theta} = \sigma\left(\sum_{v_j \in N_{v_i}^{\theta}} \alpha_{i,j}^{\theta} \cdot E_i'\right) \tag{12}$$

对于节点v_i，给定一组元路径集合$\{\theta_1, \theta_2, \cdots, \theta_N\}$，由节点级自注意力机制得到 N 个节点嵌入，表示为$\{z_{\theta_1}, z_{\theta_2}, \cdots, z_{\theta_N}\}$。为了量化每条元路径在节点嵌入表示中的权重，定义注意力权重系数β_{θ}：

$$(\beta_{\theta_1}, \beta_{\theta_2}, \cdots, \beta_{\theta_N}) = Att_{\text{sem}}(z_{\theta_1}, z_{\theta_2}, \cdots, z_{\theta_N}) \tag{13}$$

其中，Att_{sem}表示语义级注意力，用于学习每一条元路径的重要性。首先，每个节点在该元路径下的节点嵌入经过非线性转换层，获取每条元路径θ_i的权重w_{θ_i}，之后经过 softmax 归一化，获得元路径θ_i的权重表示β_{θ_i}：

$$\beta_{\theta_i} = \frac{\exp(w_{\theta_i})}{\sum_{i=1}^{N} \exp(w_{\theta_i})} \tag{14}$$

将得到的元路径系数与节点级嵌入做加权和，得到节点v_i的全局嵌入表示，如式(15)所示：

$$Z = \sum_{n=1}^{N} \beta_{\theta_n} \cdot z_{\theta_n} \tag{15}$$

最后，将经过处理和提取的特征作为输入，经过 softmax 激活函数的多层感知器 MLP 作为分类层进行分类。在训练过程中，使用交叉熵作为损失函数来衡量模型预测与真实标签之间的差异，通过最小化交叉熵损失函数准确进行漏洞分类预测。

4　实验结果及分析

4.1　智能合约数据集

本文面向 6 种常见合约漏洞进行验证，包括可重入漏洞、访问控制漏洞、算术溢出漏洞、拒绝服务漏洞等。Nguyen 等[3]构建并发布了混合的智能合约漏洞数据集，该数据集由 SmartBugs Curated[15]和 SolidiFI-BenchMark[16]构成。SmartBugs-Wild[15]数据集包含 47 398 个合约，其中 2 742 个合约不包含任何漏洞。本文使用 Slither 结合 Remix 工具的检测结果对所有合约节点类型生成标签，最终数据集包含的漏洞节点类型分布如表 1 所列。

表 1　智能合约漏洞数据集中不同种类漏洞的数据分布

Table 1　Distribution of different types of smart contract vulnerability dataset

数据	可重入漏洞	访问控制漏洞	算术逻辑漏洞	拒绝服务漏洞	前跑漏洞	时间戳漏洞
总节点数	18 898	13 770	17 372	13 968	22 824	16 765
总边数	17 614	10 721	14 271	11 997	19 761	15 550
漏洞节点数	11 238	7 500	10 110	8 280	10 008	10 051

4.2　实验环境及参数设置

本文模型基于 PyTorch 深度学习框架，开发语言为 Python，自适应学习率为[0.000 5，0.01]。为了防止过拟合，模型随机失活率 Dropout 在[0.2，0.4，0.6，0.8]内选择调整。损失函数为交叉熵损失函数(Cross Entropy Loss)。其中，对于图嵌入网络模块，模型选择 Node2vec，LINE，Metapath2vec 等经典算法进行实验，每个图嵌入层的隐藏单元大小设定为 32，其余参数根据原文献的推荐进行设置参数。对于预训练模块，分别使用 BERT，CodeBERT，GraphCodeBERT 对代码文本特征进行提取，统一设置隐藏层单元大小为 768，节点操作码文本特征的输出维度为 128。在双层注意力模型参数配置方面，统一设定节点嵌入输入维度为 128，以保持模型复杂度与性能的平衡。同时，对于模型中每个节点类型的自注意力层，每个元路径配置 8 个多头注意力机制，以增强模型对节点特征的捕捉能力。

4.3　实验结果分析

为了评估所提出模型的有效性，选择准确率(Accuracy)、精确率(Precision)、召回率(Recall)及 F1 分数作为评价指标，计算式如下：

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{16}$$

$$Precision = \frac{TP}{TP + FP} \tag{17}$$

$$Recall = \frac{TP}{TP + FN} \tag{18}$$

$$F1 = \frac{2 \times Precison \times Recall}{Precision + Recall} \tag{19}$$

其中，TP 表示实际存在漏洞的合约被正确预测的样本数量，FP 表示实际不存在漏洞的合约被错误预测为存在漏洞的样本数量，TN 表示实际不存在漏洞的合约被正确预测的样本数量，FN 表示实际存在漏洞的合约被错误预测为不存在漏洞的样本数量。

4.3.1　对比实验

为了客观地评估本文方法(MFHGA)的性能，将其与目前经典的智能合约漏洞检测方法(Manticore，Slither，Mythril)和基于图嵌入网络的方法(LINE，Node2vec，Metapath2vec)以及深度学习方法(LSTM)进行对比实验，对比方法的具体信息如下，结果如表 2 所列。

传统漏洞检测方法中：Manticore[17]基于符号执行技术进行漏洞分析；Slither[14]用于检测 Solidity 智能合约中的漏洞和安全问题；Mythril[18]则是针对以太坊智能合约的静态分析工具，用于检测安全漏洞和潜在问题。

LINE[19]方法旨在将图中的节点映射到低维向量空间，其设计目标是高效地学习大规模图数据的节点表示。

Node2vec[20]方法使用 skip-gram 调整随机游走权重，通过尽可能保留节点在图中的邻居关系和结构信息来学习节点表示。

Metapath2vec[21]结合了随机游走和词嵌入的思想，通过执行特定元路径上的随机游走生成节点序列，并学习节点的连续向量表示。

LSTM[22]是一种递归神经网络结构，旨在处理序列数据中的长期依赖关系。其在本任务中对智能合约操作码序列进行学习。

本文 MFHGA 方法是基于预训练模型与图嵌入神经网络特征融合的异构合约图注意力方法，其中，MFHGA-N，MFHGA-M 和 MFHGA-L 分别代表采用 Node2vec，Metapath2vec 及 LINE 这 3 种不同的图嵌入神经网络。

表 2 中的结果表明，MFHGA-N 在所有漏洞类型检测中表现最佳，传统漏洞检测方法的 F1 指标较低。特别是在可拒绝服务漏洞检测中，本文方法在 F1 指标上达到84.88%，相比传统深度学习检测方法 LSTM 提升了 10.62%。与 LINE 等基于图嵌入网络的基线方法相比，本文方法的 F1 指标提升了 22.34%，展现了更好的性能。这是因为本文方法结合了图嵌入神经网络及基于代码表示的预训练模型，同时考虑了智能合约图节点的网络拓扑结构信息和操作码文本语义信息，显著提升了模型性能。

表 2 不同漏洞检测方法的 F1 对比结果

Table 2 Comparison of F1 results of different vulnerability detection methods

(%)

模型	可重入漏洞	访问控制漏洞	算术逻辑漏洞	拒绝服务漏洞	前跑漏洞	时间戳漏洞
Manticore[15]	9.00	30.00	30.00	12.00	7.00	24.00
Slither[12]	15.00	32.00	45.20	13.00	26.00	44.00
Mythril[16]	19.00	34.00	73.00	41.00	63.00	23.00
LINE[17]	68.02	67.11	69.60	69.38	70.19	68.28
Node2vec[18]	56.48	54.38	58.45	53.08	49.54	56.25
Metapath2vec[19]	53.61	62.79	69.22	66.87	62.46	61.81
LSTM[20]	70.93	71.12	78.97	76.73	74.34	72.59
MFHGA-M(Ours)	77.73	77.72	79.64	81.31	80.7	80.83
MFHGA-L(Ours)	79.16	79.76	79.76	84.22	**84.37**	80.72
MFHGA-N(Ours)	**79.64**	**80.37**	**82.58**	**84.88**	83.97	**82.41**

4.3.2 消融实验

1)不同特征融合策略对性能的影响

为了比较不同特征融合策略对模型检测效果的影响，本文对比了不同维度的特征，消融实验结果如表 3 和表 4 所列。

w/o _CF 表示本文方法移除了文本特征及融合模块，仅使用图嵌入网络生成节点图结构特征$E^{(g)}$进行漏洞检测。

w/o_GNNF 表示本文方法移除了图嵌入神经网络及融合模块，仅使用节点文本特征$E^{(c)}$进行漏洞检测。

Node2vec(only)，Metapath2vec(only)和 LINE(only)表示移除本文方法的特征融合模块，分别使用 3 种不同的图嵌入网络获取特征直接进行漏洞分类检测。

当使用单一的文本特征$E^{(c)}$作为合约特征输入时，MFHGA w/o_GNNF 的整体表现相比本文模型有所下降。使用单独的图嵌入网络 Node2vec 训练得到节点特征并对其直接进行分类时，模型的性能相比 MFHGA 方法大幅下降。节点的操作码文本语义特征$E^{(c)}$对模型整体性能的贡献较小，原因在于节点操作码文本语义特征只考虑了文本序列之间的语义层面的信息，而节点的图结构特征$E^{(g)}$更多考虑了代码控制以及调用流之间的整体逻辑。整体上，融合两种维度的节点语义特征及合约图结构的节点特征对性能的提升显著。

表 3 对可重入漏洞、访问控制漏洞、算术逻辑漏洞使用不同特征融合策略的消融实验结果

Table 3 Ablation experiment results of using different feature fusion strategy for reentrancy vulnerability, access control vulnerability and arithmetic logic vulnerability

(%)

模型	可重入漏洞				访问控制漏洞				算术逻辑漏洞			
	Acc	*Pre*	*Rec*	*F1*	*Acc*	*Pre*	*Rec*	*F1*	*Acc*	*Pre*	*Rec*	*F1*
MFHGA-N(Ours)	**77.53**	**79.13**	80.16	**79.64**	**75.93**	**80.54**	80.20	**80.37**	**75.96**	**80.55**	**84.71**	**82.58**
MFHGA-M(Ours)	71.59	76.23	79.29	77.73	73.73	77.39	78.05	77.72	74.16	79.78	79.50	79.64
MFHGA-L(Ours)	74.36	74.12	**84.94**	79.16	74.73	77.66	**81.98**	79.76	75.66	76.85	82.90	79.76
MFHGA-N w/o_CF	74.66	78.37	80.89	79.61	73.87	78.73	77.74	78.23	72.34	78.92	79.12	79.02
MFHGA-M w/o_CF	69.60	72.91	67.46	70.08	66.52	68.64	61.98	65.14	65.51	61.8	67.36	64.46
MFHGA-L w/o_CF	71.37	76.04	72.45	74.2	72.45	77.5	74.27	75.85	72.82	77.71	74.63	76.14
MFHGA w/o_GNNF	70.90	75.91	72.43	74.13	72.82	75.72	78.03	76.86	73.2	77.88	76.95	77.41
Node2vec(only)	54.44	55.85	57.12	56.48	54.18	58.35	50.92	54.38	58.2	59.34	57.59	58.45
Metapath2vec(only)	58.49	59.68	48.66	53.61	60.58	60.46	65.31	62.79	58.86	61.1	79.83	69.22
LINE(only)	54.13	64.32	72.17	68.02	52.73	56.55	82.52	67.11	56.23	60.73	81.50	69.60

表 4　对拒绝服务漏洞、前跑漏洞、时间戳漏洞使用不同特征融合策略的消融实验结果

Table 4　Ablation experiment results of using different feature fusion strategy for denial of service vulnerability, front running vulnerability and timestamp vulnerability

(%)

模型	拒绝服务漏洞				前跑漏洞				时间戳漏洞			
	Acc	*Pre*	*Rec*	*F1*	*Acc*	*Pre*	*Rec*	*F1*	*Acc*	*Pre*	*Rec*	*F1*
MFHGA-N(Ours)	**82.82**	81.25	**88.85**	**84.88**	81.18	78.29	**90.54**	83.97	**79.45**	**82.06**	82.76	**82.41**
MFHGA-M(Ours)	78.12	82.09	80.54	81.31	76.72	71.03	93.42	80.7	76.44	71.53	**92.91**	80.83
MFHGA-L(Ours)	81.57	**84.94**	83.51	84.22	**82.25**	**82.22**	86.64	**84.37**	75.00	78.12	83.50	80.72
MFHGA-N w/o_CF	79.57	82.66	80.64	81.64	78.81	81.37	79.86	80.61	75.18	80.13	79.83	79.98
MFHGA-M w/o_CF	67.23	63.10	73.54	67.92	61.91	61.91	75.89	68.19	68.97	61.48	67.26	64.24
MFHGA-L w/o_CF	80.67	83.12	77.16	80.03	81.13	82.43	85.26	83.82	73.14	78.54	76.86	77.69
MFHGA w/o_GNNF	75.64	79.92	77.52	78.7	75.85	74.72	79.15	76.87	72.91	77.72	75.80	76.75
Node2vec(only)	49.17	54.75	51.51	53.08	48.09	49.77	49.31	49.54	56.18	58.09	54.52	56.25
Metapath2vec(only)	63.23	70.11	63.92	66.87	59.04	62.41	62.51	62.46	58.91	63.21	60.47	61.81
LINE(only)	56.67	55.49	92.55	69.38	56.38	63.35	78.69	70.19	54.18	61.59	76.60	68.28

2)使用不同预训练模型对模型性能的影响

本文设计实验使用不同预训练模型的代码文本特征生成模块，验证其对模型性能的影响。实验结果如表 5 和表 6 所列，表中 FHGA 为本文提出的双层异构图注意力融合模块。

表 5　对可重入漏洞、访问控制漏洞、算术逻辑漏洞使用不同预训练模型生成文本特征的实验结果

Table 5　Experimental results of using different pre-trained models to generate text features for reentrancy vulnerability, access control vulnerability and arithmetic logic vulnerability

(%)

模型	可重入漏洞				访问控制漏洞				算术逻辑漏洞			
	Acc	*Pre*	*Rec*	*F1*	*Acc*	*Pre*	*Rec*	*F1*	*Acc*	*Pre*	*Rec*	*F1*
MFHGA-M+CodeBERT(Ours)	71.59	75.23	80.40	77.73	73.73	77.39	78.05	77.72	74.16	79.78	79.50	79.64
MFHGA-L+CodeBERT(Ours)	74.36	74.12	**84.94**	79.16	74.73	77.66	**81.98**	79.76	75.66	76.85	82.90	79.76
MFHGA-N+CodeBERT(Ours)	**77.53**	**79.13**	80.16	**79.64**	**75.93**	**80.54**	80.20	**80.37**	**75.96**	**80.55**	**84.71**	**82.58**
FHGA-M +BERT	70.71 (−0.88)	76.07 (+0.84)	77.50 (−2.90)	76.78 (−0.95)	71.34 (−2.39)	76.24 (−1.15)	75.76 (−2.29)	76.00 (−1.72)	73.28 (−0.88)	79.51 (−0.27)	77.48 (−2.02)	78.48 (−1.16)
FHGA-L +BERT	73.45 (−0.91)	74.75 (−0.63)	81.52 (−3.41)	77.99 (−1.17)	73.52 (−1.21)	70.11 (−7.55)	81.37 (−0.61)	75.32 (−4.44)	74.23 (−1.43)	76.95 (+0.10)	79.93 (−2.97)	78.41 (−1.35)
FHGA-N +BERT	74.62 (−2.91)	71.27 (−7.86)	82.93 (+2.78)	76.66 (−2.98)	74.91 (−1.02)	75.60 (−4.94)	76.73 (−3.47)	76.16 (−4.21)	74.69 (−1.27)	76.51 (−4.04)	80.55 (−4.16)	78.48 (−4.10)
FHGA-M +GraphCodeBERT	72.13 (+0.54)	76.68 (+1.45)	69.73 (−10.6)	73.04 (−4.69)	69.95 (−3.78)	72.21 (−5.18)	69.81 (−8.24)	70.99 (−6.73)	70.45 (−3.71)	74.78 (−5.00)	68.33 (−11.2)	71.41 (−8.23)
FHGA-L +GraphCodeBERT	72.70 (−1.66)	71.18 (−2.94)	82.96 (−1.98)	76.62 (−2.54)	72.30 (−2.43)	77.30 (−0.36)	81.05 (−0.93)	79.13 (−0.63)	74.10 (−1.56)	76.01 (−0.84)	78.55 (−4.35)	77.26 (−2.50)
FHGA-N +GraphCodeBERT	76.52 (−1.01)	76.21 (−2.92)	78.90 (−1.26)	77.53 (−2.11)	76.22 (−0.29)	78.48 (−2.06)	76.66 (−3.54)	77.56 (−2.81)	75.63 (−0.33)	81.37 (−0.82)	79.47 (−5.24)	80.41 (−2.17)

表 6　对于拒绝服务漏洞、前跑漏洞、时间戳漏洞使用不同预训练模型生成文本特征的实验结果

Table 6　Experimental results of using different pre-trained models to generate text features for denial of service vulnerability, front running vulnerability and timestamp vulnerability

(%)

模型	拒绝服务漏洞				前跑漏洞				时间戳漏洞			
	Acc	*Pre*	*Rec*	*F1*	*Acc*	*Pre*	*Rec*	*F1*	*Acc*	*Pre*	*Rec*	*F1*
MFHGA-M+CodeBERT(Ours)	78.12	82.09	81.33	81.71	76.72	71.03	93.42	80.70	76.44	71.53	92.91	80.83
MFHGA-L+CodeBERT(Ours)	81.57	**84.94**	81.57	**83.22**	**82.25**	**82.22**	86.64	**84.37**	75.00	78.12	83.50	80.72
MFHGA-N+CodeBERT(Ours)	**82.82**	81.25	**82.52**	81.88	81.18	78.29	**90.54**	83.97	**79.45**	82.06	82.76	**82.41**
FHGA-M+BERT	76.41 (−1.71)	71.47 (−10.6)	78.19 (−3.14)	74.68 (−7.03)	75.05 (−1.67)	70.17 (−0.86)	89.05 (−4.37)	78.49 (−2.21)	74.32 (−2.12)	70.63 (−0.90)	88.57 (−4.34)	78.59 (−2.24)
FHGA-L+BERT	80.77 (−0.80)	78.89 (−6.05)	81.55 (−0.01)	80.20 (−3.02)	81.22 (−1.03)	76.53 (−5.69)	89.48 (+2.84)	82.50 (−1.87)	75.92 (+0.92)	72.41 (−5.71)	84.64 (+1.14)	78.05 (−2.67)
FHGA-N+BERT	80.88 (−1.94)	77.43 (−3.82)	80.47 (−2.05)	78.92 (−2.96)	79.18 (−2.00)	75.30 (−2.99)	82.95 (−7.59)	78.94 (−5.03)	77.19 (−2.26)	74.91 (−7.15)	88.52 (−5.76)	81.15 (−1.26)
FHGA-M+GraphCodeBERT	71.67 (−6.45)	75.97 (−6.12)	69.87 (−1.47)	72.79 (−8.92)	72.13 (−4.59)	71.68 (+0.65)	85.64 (−7.78)	78.04 (−2.66)	71.09 (−5.35)	75.31 (+3.78)	83.36 (−9.55)	79.13 (−1.70)
FHGA-L+GraphCodeBERT	81.94 (+0.37)	82.48 (−2.46)	78.21 (−3.35)	80.29 (−2.93)	80.85 (−1.40)	82.67 (+0.45)	79.57 (−7.07)	81.09 (−3.28)	73.09 (−1.91)	77.70 (−0.42)	72.61 (−10.89)	75.07 (−5.65)
FHGA-N+GraphCodeBERT	80.47 (−2.35)	80.86 (−0.39)	82.33 (−.019)	81.59 (−0.29)	79.35 (−1.83)	78.87 (−0.58)	81.41 (−9.13)	80.12 (−3.85)	77.21 (−2.24)	80.36 (−1.70)	81.98 (−0.79)	81.16 (−1.25)

使用 CodeBERT 作为节点文本特征模块时，整体模型 MFHGA 表现最佳；使用 BERT 模型进行特征表示时，相比

于 CodeBERT 模型性能略差；使用 GraphCodeBERT 作为节点文本特征生成的预训练模型时，模型整体效果下降幅度较大。本文提出的合约图节点特征融合方法中，获取了图语义节点特征，而 GraphCodeBERT 预训练模型重复考虑该类特征，忽视了其他重要的特征，导致信息存在很高的相关性，模型效果表现不佳。使用不同的文本生成模型的实验结果表明，本文方法的性能表现最优。

3）去噪操作对模型性能的影响

本文选取了可重入漏洞、访问控制漏洞和算术逻辑漏洞 3 种漏洞进行去噪操作的消融实验，以评估去噪操作在去除智能合约中的冗余或无关信息方面的有效性，以及探讨去噪处理后对漏洞检测性能的具体提升程度。

实验结果如表 7 所列，采用去噪技术后，模型的准确率提升了 2.29～5.72 个百分点，F1 指标提升 5.72～10.12 个百分点，性能显著提升，表明进行噪声处理后模型更能聚焦关键信息。整体而言，去噪方法在清除冗余信息的同时保留了关键数据，有效提高了数据质量，增强了模型的漏洞检测能力，为智能合约安全分析提供了可靠的支持。

表 7 对于可重入漏洞、访问控制漏洞、算术逻辑漏洞是否去噪处理的消融实验

Table 7 Ablation experiment on whether to denoise for reentrancy vulnerability, access control vulnerability and arithmetic logic vulnerability

(%)

模型	去噪处理	可重入漏洞			访问控制漏洞			算术逻辑漏洞		
		Pre	*Rec*	*F1*	*Pre*	*Rec*	*F1*	*Pre*	*Rec*	*F1*
MFHGA-N(Ours)	×	71.79	74.81	73.27	73.32	75.55	74.42	71.52	77.06	74.19
MFHGA-M(Ours)	×	66.20	78.93	72.01	68.90	72.76	70.78	70.02	72.83	71.40
MFHGA-L(Ours)	×	68.64	72.62	70.57	72.44	69.50	70.91	70.46	68.83	69.64
MFHGA-N(Ours)	√	77.53	80.16	79.64	75.93	80.20	80.37	75.96	84.71	82.58
MFHGA-M(Ours)	√	71.59	79.29	77.73	73.73	78.05	77.72	74.16	79.50	79.64
MFHGA-L(Ours)	√	74.36	84.94	79.16	74.73	81.98	79.76	75.66	82.90	79.76

结束语 本文提出了一种基于异构合约图多维度特征深度融合的漏洞检测方法，通过在智能合约中有效提取操作码字符的语义特征以及合约图的结构特征，采用图神经异构网络注意力机制实现多维特征融合。将预训练语言模型与图嵌入网络联合应用于智能合约代码的分析，捕获更加丰富的特征信息，提升漏洞检测性能。使用双层注意力机制结合元路径信息对所提取的图特征以及语义特征进行深度融合，更准确地识别出潜在的安全漏洞。本文对使用不同预训练模型来获取语义特征的效果进行了深入对比分析，与经典智能合约漏洞检测方法及基线模型的比较结果表明，本文模型具有较好的检测性能。

在实际场景中，本文方法可以集成到智能合约应用的各个阶段中：在开发阶段，可将本文方法作为漏洞检测插件集成到开发工具中，实时对合约代码的安全性检查，识别本模型所覆盖的常见漏洞，如可重入漏洞、整数溢出漏洞等，减少合约代码开发过程中可能产生的风险；在审计阶段，可与现有的审计框架集成，帮助审计员减少重复性工作，快速定位合约中的潜在问题，节省审计时间。

在未来的研究中，将进一步提升模型的鲁棒性，通过 GAN 网络生成对抗样本，同时扩大数据集规模，增加静态漏洞种类以外的动态复杂漏洞；结合对抗多任务学习，生成对抗样本，共享复杂漏洞共有特征，提升模型的对抗性防御能力；引入大型预训练语言模型参与训练学习，利用模型在语言理解方面的能力，进一步提升对智能合约代码的语义分析能力，提高检测准确率。

参 考 文 献

[1] DONG W L, LIU Z, LIU K, et al. Survey on Vulnerability Detection Technology of Smart Contracts[J]. Journal of Software, 2023, 35(1): 38-62.

[2] ZHANG Y L, MA J L, LIU Z A, et al. A Survey of vulnerability detection methods for Ethereum Solidity smart contracts[J]. Computer Science, 2022, 49(3): 52-61.

[3] NGUYEN H H, NGUYEN N M. MANDO-HGT: Heterogeneous Graph Transformers for Smart Contract Vulnerability Detection[C]// 2023 IEEE/ACM 20th International Conference on Mining Software Repositories(MSR). IEEE, 2023: 334-346.

[4] PARVEEN N, CHAKRABARTI P, HUNG B T, et al. Twitter sentiment analysis using hybrid gated attention recurrent network[J]. Journal of Big Data, 2023, 10(1): 50.

[5] CAI J, LI B, ZHANG J, et al. Combine sliced joint graph with graph neural networks for smart contract vulnerability detection [J]. Journal of Systems and Software, 2023, 195: 111550.

[6] LUO F, LUO R, CHEN T, et al. Scvhunter: Smart contract vulnerability detection based on heterogeneous graph attention network[C]// Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. 2024: 1-13.

[7] PASQUA M, BENINI A, CONTRO F, et al. Enhancing Ethereum smart-contracts static analysis by computing a precise Control-Flow Graph of Ethereum bytecode[J]. Journal of Systems and Software, 2023, 200: 111653.

[8] ZHEN Z, ZHAO X, ZHANG J, et al. DA-GNN: A smart contract vulnerability detection method based on Dual Attention Graph Neural Network[J]. Computer Networks, 2024, 242: 110238.

[9] SHAHBAZ M, SURESH L, REXFORD J, et al. Elmo: Source routed multicast for public clouds[C]// Proceedings of the ACM Special Interest Group on Data Communication. 2019: 458-471.

[10] WANG B, XIE Q, PEI J, et al. Pre-trained language models in biomedical domain: A systematic survey[J]. ACM Computing Surveys, 2023, 56(3): 1-52.

[11] DEVLIN J, CHANG M W, LEE K, et al. BERT: pre-training of deep bidirectional transformers for language understanding [C]// Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics:

Human Language Technologies. Stroudsburg, PA: ACL, 2019: 4171-4186.

[12] GUO D, REN S, LU S, et al. GraphCodeBERT: Pre-training Code Representations with Data Flow[C]// International Conference on Learning Representations. 2021.

[13] ZHANG F, GUO D, TANG D, et al. CodeBERT: A Pre-Trained Model for Programming and Natural Languages[C]// Findings of the Association for Computational Linguistics: EMNLP. 2020: 1536-1547.

[14] 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 (WETSEB). IEEE, 2019: 8-15.

[15] FERREIRA J F, CRUZ P, DURIEUX T, et al. Smartbugs: A framework to analyze solidity smart contracts[C]// Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering. 2020: 1349-1352.

[16] GHALEB A, PATTABIRAMAN K. How effective are smart contract analysis tools? evaluating smart contract static analysis tools using bug injection[C]// Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. 2020: 415-427.

[17] MOSSBERG M, MANZANO F, HENNENFENT E, et al. Manticore: A user-friendly symbolic execution framework for binaries and smart contracts[C]// 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2019: 1186-1189.

[18] 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. 2020: 530-541.

[19] CHOUDHARY K, DECOST B. Atomistic line graph neural network for improved materials property predictions[J]. Computational Materials, 2021, 7(1): 185.

[20] ZHANG Y C, TANG M. A Theoretical Analysis of DeepWalk and Node2vec for Exact Recovery of Community Structures in Stochastic Blockmodels[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023, 46(2): 1065-1078.

[21] DONG Y, CHAWLA N V, SWAMI A. metapath2vec: Scalable representation learning for heterogeneous networks[C]// Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2017: 135-144.

[22] YU Y, SI X, HU C, et al. A review of recurrent neural networks: LSTM cells and network architectures[J]. Neural computation, 2019, 31(7): 1235-1270.

ZHOU Tao, born in 2000, postgraduate. Her main research interests include deep learning and smart contract vulnerability detection.

HAN Honggui, born in 1983, professor, Ph.D supervisor. His main research interests include machine learning and artificial intelligence.

（责任编辑：何杨）