多模态深度学习在源代码漏洞检测中的应用研究

黄 蓉[1] 吴 曦[1]
HUANG Rong WU Xi

摘 要 针对源代码静态分析中存在的语义建模不足与结构感知能力弱等问题，文章提出一种基于多模态深度学习的源代码漏洞检测方法。该方法融合 Token 序列、抽象语法树（AST）与控制流图（CFG）三类信息，分别通过 Transformer 与图神经网络提取模态特征，并引入注意力机制实现语义对齐与特征融合。在 Juliet 与 Devign 两个主流数据集上开展性能验证，并设计对比实验与消融实验评估模型有效性。结果表明，所提方法在准确率、召回率与 F_1 值方面均优于单模态或双模态模型，具备良好的检测精度与泛化能力，为深度学习在源代码静态漏洞分析中的实用化提供了新路径。

关键词 多模态学习；深度漏洞检测；抽象语法树；控制流图；图神经网络

doi：10.3969/j.issn.1672-9528.2025.12.008

0 引言

随着软件系统在操作系统内核、金融支付与工业控制等关键领域的广泛部署，源代码中潜在的安全漏洞问题日益突出，已成为影响系统稳定性与信息安全的主要隐患。传统静态分析方法以规则匹配与手工特征为主，难以应对现代源代码中复杂的语法变异与路径依赖关系，存在覆盖范围有限与误报率高的问题 [1]。近年来，深度学习技术在程序分析任务中展现出良好效果，尤其多模态学习方法在融合语义、语法与控制结构等信息方面具有显著优势 [2]。针对现有方法中语义理解能力不足、结构表示不全面的问题，本文构建一种融合 Token 序列、抽象语法树（AST）与控制流图（CFG）的多模态深度学习模型，通过图神经网络与 Transformer 等机制实现结构感知与上下文理解的统一建模，旨在提升源代码漏洞检测的准确性与鲁棒性，并验证其在标准数据集与实际样本中的实用价值。

1 源代码漏洞特征与多模态建模分析

1.1 常见漏洞类型与数据结构特征

源代码漏洞是指程序在编译或运行前即潜伏存在的结构性缺陷，常由不安全的内存操作、控制流设计不当或输入验证缺失引发。典型漏洞如缓冲区溢出、空指针解引用、SQL 注入、命令注入等，在静态层面通常表现为对数据边界、资源释放、访问权限等约束条件的违反 [3]。这些漏洞并非孤立存在，而是依附于程序内部的复杂语法结构和控制依赖路径之中，具有明显的语义依赖性与路径敏感性。以空指针解引用为例，其形成路径往往横跨多个函数作用域与条件语句分支，仅凭局部特征难以准确识别。这决定了漏洞检测任务必须对源代码进行全结构建模，捕捉其控制流与数据流交织下的隐含异常模式。

1.2 源代码多模态表示方式

为了有效建模漏洞关联的多层次特征，本文采用多模态方法对源代码进行联合表示，主要包括文本模态（Token 序列）、语法模态（抽象语法树 AST）和控制模态（控制流图或数据流图）。Token 序列保留源代码的词法与命名信息，是语义上下文建模的基础；AST 反映程序的层级结构与嵌套语法关系；CFG/DFG 则建模执行路径与变量传播依赖，是理解路径敏感漏洞不可或缺的维度。

为统一多模态输入结构，本文构建三元组形式的代码表示 [4]：

$$M = \{T, A, C\} \tag{1}$$

式中：$T=\{t_1,t_2,\cdots,t_n\}$ 表示源代码 Token 序列；$A=(V_A, E_A)$ 表示 AST 图结构；$C=(V_C, E_C)$ 表示控制流或数据流图。

各模态在处理过程中需通过语义对齐机制保持一致性，实现跨模态的节点映射与上下文联动。以函数调用为例，其 Token 表示提供语义内容；AST 表示调用结构；CFG 表示其可能路径，这些信息只有在联合建模下才能完整捕捉。

1.3 多模态信息融合需求分析

源代码中的不同模态信息具有显著互补性，单一模态模型难以完整刻画漏洞触发机制。文本模态擅长捕捉局部上下文语义，缺乏结构建模能力；而语法与控制模态可表达路径依赖与程序结构，却在命名语义与语言理解方面存在不足。

1. 信阳艺术职业学院 河南信阳 464000

因此，漏洞检测需构建多模态融合机制[5]，联合三类信息流提升语义解析与结构感知能力。融合建模应满足以下需求：实现 Token、AST 节点与控制图节点的语义对齐；设计可训练的融合模块以动态调节模态权重；通过端到端网络结构完成统一表示与分类判别。后续模型将基于多分支编码器、图神经网络与注意力机制满足上述要求，实现对复杂漏洞模式的精准识别。

2 多模态深度学习模型构建

2.1 模型总体架构设计

针对源代码漏洞检测中存在的语义、结构与路径异构信息融合问题，本文设计了一种基于多模态深度融合的端到端神经网络架构。该模型采用多分支特征提取 + 模态交互融合 + 联合判别输出的设计思想，能够协同处理 Token 序列、AST 图结构与 CFG 控制路径信息，实现多源语义的深层建模与交叉感知[6]。

模型输入为一个三元组 $M = \{T, A, C\}$，分别代表源代码的 Token 序列、抽象语法树与控制流图。每种模态通过独立的特征提取子网络编码为中间表示向量，随后在融合模块中通过注意力机制与残差映射策略进行联合建模，最终通过全连接分类器输出是否存在漏洞的概率标签。

该结构可形式化表示如下：

（1）输入阶段：Token、AST 与 CFG 分别编码为嵌入张量；

（2）提取阶段：$\boldsymbol{H}_T = f_T(T)$，$\boldsymbol{H}_A = f_A(A)$，$\boldsymbol{H}_C = f_C(C)$；

（3）融合阶段：$\boldsymbol{H} = \text{Fusion}(\boldsymbol{H}_T, \boldsymbol{H}_A, \boldsymbol{H}_C)$；

（4）判别阶段：$\hat{y} = \sigma(\boldsymbol{W}\boldsymbol{H} + b)$，其中 $\hat{y} \in [0,1]$。

该模型架构的核心目标是在保持模态独立性结构优势的同时，实现特征语义空间的一致性重建，为漏洞判别提供高维统一表达基础。

2.2 模态特征提取网络设计

不同模态具有不同的结构特点与信息密度，因此本文在特征提取阶段采用定制化神经网络结构分别处理三种模态。

（1）Token 序列建模子网络

文本模态采用预训练 Transformer 结构进行上下文建模。首先将每个 Token 通过词嵌入层映射至高维语义向量，再通过多层 Transformer 编码器捕捉上下文依赖。该网络具备较强的语言建模能力，适合用于函数名、变量名、注释等自然语言成分的深层语义表示[7]。

$$\boldsymbol{H}_T = \text{TransformerEncoder}(\text{Embed}(T)) \tag{2}$$

（2）AST 结构建模子网络

语法模态建模采用图神经网络（graph neural networks, GNN），如（gated graph neural network, GGNN）或（graph sample and aggregate, GraphSAGE）。AST 被表示为节点属性与边类型的组合图，通过消息传递机制在图结构上传递语法信息，形成局部语法子树到全图的逐层聚合表示。

$$\boldsymbol{H}_A^{(l+1)} = \sigma\left(W_A \cdot \text{AGG}\left(\{\boldsymbol{H}_u^{(l)} : u \in N(v)\}\right) + b_A\right) \tag{3}$$

式中：$N(v)$ 表示节点 v 的邻接节点集合；AGG 表示聚合函数（如 mean 或 attention-based pooling）。

（3）控制流建模子网络

CFG 或 DFG 结构同样采用图神经网络建模，但更注重路径方向性与边权权重的处理[8]。本文选用带方向增强机制的 GAT（Graph Attention Network），为控制流图中每条边分配注意力权重，捕捉关键路径与依赖关系。

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}(a \cdot [Wh_i \,\|\, Wh_j])\right)}{\sum_{k \in N(v)} \exp\left(\text{LeakyReLU}(a \cdot [Wh_i \,\|\, Wh_k])\right)} \tag{4}$$

最终输出 $\boldsymbol{H}_C$ 表示为带加权路径注意力聚合的控制模态向量。

2.3 模态融合与联合表示机制

在完成 3 个模态的特征提取后，需对其进行统一语义空间的融合处理[4]。本文融合策略包括线性融合、模态注意力机制与残差增强连接，以增强特征间互补性。

首先，对每个模态输出进行维度映射，投影至统一维度空间 $\mathbf{R}^d$，表示为：

$$\tilde{\boldsymbol{H}}_T = W_T \boldsymbol{H}_T, \quad \tilde{\boldsymbol{H}}_A = W_A \boldsymbol{H}_A, \quad \tilde{\boldsymbol{H}}_C = W_C \boldsymbol{H}_C \tag{5}$$

然后，通过模态注意力机制分配不同模态的权重：

$$\boldsymbol{\alpha}_m = \frac{\exp(u \cdot \tanh(W_m \tilde{\boldsymbol{H}}_m + b_m))}{\sum_{k \in \{T,A,C\}} \exp(u \cdot \tanh(W_k \tilde{\boldsymbol{H}}_k + b_k))} \tag{6}$$

最终联合表示向量为：

$$\boldsymbol{H} = \sum_{m \in \{T,A,C\}} \alpha_m \cdot \tilde{\boldsymbol{H}}_m + \text{Residual}([\tilde{\boldsymbol{H}}_T, \tilde{\boldsymbol{H}}_A, \tilde{\boldsymbol{H}}_C]) \tag{7}$$

该融合结构兼具信息聚合与分模态解释能力，支持不同类型漏洞触发路径的语义适配与结构识别。融合后的表示将送入多层感知器进行二分类判别，输出漏洞存在的概率标签，训练过程中使用二元交叉熵作为损失函数进行优化。

3 数据集构建与处理方法

3.1 数据集选取与标注规范

高质量的数据集是训练多模态漏洞检测模型的基础。本文选用 Juliet Test Suite (C/C++) 与 Devign 数据集作为主要训练与验证样本来源，前者由美国国家标准与技术研究院（NIST）维护，覆盖 CWE 定义的多种典型漏洞类型，后者为 GitHub 真实开源项目中的函数级代码片段，并配有人工标注的漏洞标签[9]。

为确保样本的一致性与可融合性，统一以函数级源代码片段作为基本分析单元，每个样本包含：源代码文本、漏洞

标签、语法结构及控制路径信息。标注策略遵循如下规范：

（1）每条样本标记二分类标签 $y\in\{0,1\}$，其中 1 表示存在 CWE 定义的可利用漏洞；0 表示无安全风险。

（2）多漏洞样本按“存在即正类”原则标记为 1。

（3）所有样本均通过静态分析工具（如 Flawfinder、Cppcheck）与人工复核联合完成标注，确保准确性与一致性。

此外，对不满足最小 Token 数量（小于 10）或解析失败的异常样本进行剔除处理，最终构建包含约 32 000 条函数片段的高质量数据集，覆盖 20 类以上主流漏洞类型。

3.2 静态分析与特征提取流程

为支持多模态建模，需从原始源代码中解析出多维结构化特征，建立完整的语法树、控制依赖与语义信息表达。处理流程包括词法预处理、结构解析与图构建三阶段，具体如下：

（1）Token 抽取：采用 ANTLR 语法工具对源代码进行词法分析，保留关键标识符、操作符、关键词与注释，形成 Token 序列 $T=\{t_1,t_2,...,t_n\}$，同时记录位置信息与词性标签，用于后续位置嵌入与语义对齐。

（2）抽象语法树生成：使用 Clang AST API 对每个函数片段构建完整抽象语法树（AST），节点包括语句、表达式、类型信息等，边表示语法嵌套与父子结构关系。树结构转为图表示 $A=(V_A, E_A)$，供 GNN 子网络输入。

（3）控制流 / 数据流图提取：基于 LLVM IR 中间码构建控制流图（CFG）与数据流图（DFG），节点对应基本块或变量操作，边表示执行路径与依赖关系。图结构统一为 $C=(V_C, E_C)$，并进行边类型与跳转方向的显式编码处理 [10]。

所有静态特征抽取均采用自动化流程批量处理，避免人工干预，提高数据一致性。为降低图规模差异对训练的干扰，图结构在节点数量超过阈值时进行随机裁剪与边重采样，以增强训练稳定性。

3.3 模态对齐与样本生成

由于各模态数据在结构维度与语义编码方式上存在异构性，为实现联合建模，需建立模态间的一致性对齐机制，并构建规范化的样本输入格式。本文采用以下对齐策略：

（1）Token-AST 对齐：通过 AST 节点的源代码位置字段与 Token 位置信息匹配，建立语义锚点映射，确保每个 Token 在 AST 中能唯一定位至其父语法单元；

（2）Token-CFG 对齐：将 CFG 基本块内的操作映射回 Token 序列，根据词法位移建立跳转点与控制依赖之间的语义连接；

（3）跨模态节点映射表：为每个样本构建跨模态索引矩阵 $\boldsymbol{M}_{\text{align}}\in\mathbf{R}^{|T|\times(|V_A|+|V_C|)}$，用于训练阶段的信息传递与注意力融合。

样本最终组织为统一输入结构 $\{T, A, C, y\}$ 即：Token 序列 T、AST 图结构 A、控制图结构 C 与标签 y。所有样本经规范化处理后保存为 JSONL 格式，便于后续加载与批处理训练。

该处理流程保证三模态在结构空间与语义空间上的一致性，为深度融合模型提供了结构完备、对齐精确的训练基础。

4 实验设计与评估分析

4.1 实验设置与指标定义

为验证所构建多模态深度学习模型在源代码漏洞检测任务中的有效性，在高性能实验环境中开展训练与评估。实验平台采用 NVIDIA RTX 3090 GPU（24 GB 显存）、Intel Core i9-13900K 处理器与 128 GB DDR5 内存，操作系统为 Ubuntu 20.04 LTS，深度学习框架为 PyTorch 2.0，Python 版本为 3.10。图处理部分集成 DGL（Deep Graph Library），静态分析工具链基于 LLVM 14.0 与 Clang AST。模型训练采用 AdamW 优化器，初始学习率设置为 1e-4，batch size 为 32，训练轮次 100 轮，EarlyStopping 阈值为 5 轮。

性能评估采用四项主流指标：准确率（Accuracy）、精确率（Precision）、召回率（Recall）与 F_1 值（F_1-score），以综合衡量模型在漏洞检测任务中对正负样本的识别能力与误报控制能力。

4.2 多模态模型性能验证

为验证所提模型在多模态联合建模下的性能优势，分别在 Juliet 与 Devign 两个数据集上进行训练与测试，采用 10% 样本划分为验证集，其余按 8∶2 划分为训练集与测试集。表 1 展示了模型在两个数据集上的主评估指标。

表 1 多模态模型在不同数据集下的检测性能指标

单位：%

数据集	Accuracy	Precision	Recall	F_1-score
Juliet	94.60	92.10	91.30	91.70
Devign	92.80	89.50	87.90	88.70

表 1 结果显示，模型在 Juliet 数据集上取得 94.6% 的总体准确率，精确率与召回率分别达到 92.1% 与 91.3%，F_1 值为 91.7%，表现出对多类型静态漏洞的稳定检测能力；在真实开源项目构成的 Devign 数据集上，模型同样保持 92.8% 的准确率与 88.7% 的 F_1-score，表明其对实际复杂语义结构具有良好适应性。结果验证了三模态特征联合学习在结构捕捉与上下文理解方面的综合优势。

4.3 对比与消融实验分析

为进一步评估多模态设计的有效性，设置对比实验与消融实验，分别分析单模态与模态组合对检测性能的影响。如表 2 所示。

表 2 模态结构对比与消融实验结果（Juliet 数据集）

单位：%

模型结构	Accuracy	Precision	Recall	F_1-score
Token-only	85.30	82.00	78.90	80.40
AST-only	87.10	83.50	80.70	82.10
CFG-only	86.40	84.10	76.80	80.20
Token + AST	90.60	88.20	85.00	86.60
Token + CFG	89.70	87.60	83.10	85.30
AST + CFG	88.90	85.00	82.40	83.70
Token + AST + CFG	94.60	92.10	91.30	91.70

从表 2 结果可见，单模态结构整体性能明显低于任意双模态组合。Token-only 模型虽在文本语义识别上表现稳定，但缺乏结构感知能力，导致 Recall 偏低；AST-only 与 CFG-only 模型可建模结构关系，但在语义细节理解方面存在不足。双模态组合提升明显，尤其是 Token+AST 组合在语法控制与上下文建模间取得良好平衡。三模态融合结构在四项指标上均达到最优，F_1 提升幅度超过 11%，验证了语义信息、语法结构与控制依赖在联合学习下具备显著互补性。

4.4 源代码漏洞检测实证分析

为进一步验证多模态深度学习模型在真实场景中的漏洞检测能力，本文选取 Juliet 与 Devign 数据集中三类高频漏洞（缓冲区溢出、空指针解引用、SQL 注入）进行针对性实验。对每类漏洞分别构建专属子集，评估模型对特定漏洞模式的识别精度与误报率，以验证其在细粒度检测场景中的稳定性与适用性。实验结果如表 3 所示。

表 3 不同漏洞类型下多模态模型检测性能评估结果

单位：%

漏洞类型	准确率	召回率	精确率	F_1-score
缓冲区溢出	95.2	93.7	94.6	94.1
空指针解引用	93.6	91.8	92.3	92
SQL 注入	91.8	88.4	90.2	89.3

从表 3 可见，模型在缓冲区溢出与空指针解引用检测中表现出较高的识别精度与鲁棒性，说明融合 Token、AST 与 CFG 模态的深层建模策略在结构异常与路径异常识别中具备实际可行性。在 SQL 注入等语义敏感型漏洞中，模型虽保持良好性能，但表现略低，提示需在未来工作中进一步增强语言语义的上下文理解能力。

此外，本文还选取真实 CVE 样本（如 CVE-2017-5638，CVE-2019-11043）进行模型实测，发现模型在无需显示规则输入的情况下能正确识别关键缺陷路径，进一步验证其对实战场景的适配能力。

5 结论

本文构建了一种融合 Token、AST 与 CFG 三种模态表示的源代码漏洞检测模型，采用 Transformer 与图神经网络提取不同信息维度的特征，并通过注意力机制实现语义融合与统一判别。在 Juliet 与 Devign 两个标准数据集上进行的实验证明，该方法在 F_1 值与准确率方面均显著优于传统单模态与双模态模型，表现出良好的跨样本鲁棒性与检测能力。消融实验进一步验证多模态结构对复杂语义与路径依赖建模的有效贡献。此外，在具体漏洞类型的细粒度评估中，模型在缓冲区溢出、空指针解引用等典型漏洞识别上展现出较强性能，具备实际工程适配潜力。未来工作将重点聚焦于语义感知精度提升与跨语言漏洞迁移能力拓展，以推动多模态学习在软件安全领域的广泛应用。

参考文献：

[1] 于巧，黄子睿，程圣懿，等．基于边权重的软件漏洞检测方法 [J/OL]. 计算机应用，1-10[2025-10-17].https://link.cnki.net/urlid/51.1307.TP.20250519.1734.006.

[2] 王春东，牛德合，苗春雨．多级别智能合约漏洞检测与定位 [J/OL]. 天津理工大学学报，1-10[2025-10-25].https://link.cnki.net/urlid/12.1374.n.20250520.1014.004.

[3] 曲豫宾，黄松，陈翔，等．面向深度漏洞检测模型的黑盒对抗攻击 [J]. 软件学报，2025,36(11):5062-5081.

[4] 陈旭，陈子雄，景永俊，等．基于双曲图卷积神经网络的切片级漏洞检测方法 [J]. 计算机工程与科学，2025, 47(5): 851-863.

[5] 张学军，郭梅凤，张潇，等．基于深度学习的混合语言源代码漏洞检测方法 [J]. 湖南大学学报（自然科学版），2025, 52(4): 103-113.

[6] 曹思聪，孙小兵，薄莉莉，等．基于结构感知图神经网络的多类别漏洞检测 [J]. 软件学报，2025,36(11):5045-5061.

[7] 唐成华，张靖，杨萌萌，等．结合思维链提示与 Conformer 的软件漏洞检测方法 [J/OL]. 计算机工程与应用，1-13[2025-10-17].https://link.cnki.net/urlid/11.2127.TP.20250312.1618.020.

[8] 陈旭，陈子雄，景永俊，等．一种融合双曲表示与欧几里得表示的源代码漏洞检测方法 [J/OL]. 郑州大学学报（理学版），1-8[2025-10-13].https://doi.org/10.13705/j.issn.1671-6841.2024119.

[9] 李雯洁，李雷孝，刘东江，等．数据交易中智能合约漏洞检测研究综述 [J]. 计算机工程与应用，2025,61(9):1-24.

[10] 陈子雄，陈旭，景永俊，等．基于图神经网络的源代码漏洞检测研究综述 [J]. 计算机工程与科学，2024, 46(10): 1775-1792.

【作者简介】

黄蓉（1998—），女，河南淮滨人，硕士，助教，研究方向：边缘计算、人工智能研究。

（收稿日期：2025-06-18 修回日期：2025-11-28）