基于异构特征协同的轻量化恶意软件分类方法

陈治国[1]，赵攀[1]，谢雪[2]，郑文鑫[3]，王金伟[4]

（1.南京信息工程大学计算机学院、网络空间安全学院，江苏 南京 210044；2.中国航天系统科学与工程研究院，北京 100037；3.奇安信科技集团股份有限公司 盘古事业部，北京 100044；4.南开大学密码与网络空间安全学院，天津 300071）

摘 要：针对静态恶意软件分类中单一特征覆盖信息不足，以及多特征融合模型计算复杂度高、部署受限的问题，本文提出一种面向资源受限场景的轻量化多特征协同建模方法，结合操作码序列与静态 API 调用信息，从指令执行结构与系统交互两个维度刻画恶意软件行为特征。在操作码分支中，采用多尺度残差建模以强化对局部指令模式和长程依赖的表征能力；针对 API 高维特征稀疏与冗余问题，在 API 分支提出三元组语义链表示方法，将离散调用细化为结构化功能信息，增强高层功能语义表达和冗余抑制；在分类阶段引入位置感知多头自注意力机制，对两类特征进行动态融合与联合建模，在保证分类精度的同时有效控制模型规模。实验表明，该方法在 BIG2015 和企业级 PE 数据集上分别达到 99.53% 和 99.4% 的准确率，模型总浮点运算量为 30.2 MFLOPs，为资源受限场景下的恶意软件分类提供了高精度、低开销的解决方案。

关键词：恶意软件分类；深度学习；轻量化；操作码；API

中图分类号：TP309

Lightweight malware classification method based on heterogeneous feature collaboration

CHEN Zhiguo[1], ZHAO Pan[1], XIE Xue[2], ZHENG Wenxin[3], WANG Jinwei[4]

1. School of Computer, Nanjing University of Information Science and Technology, Nanjing 210044, China
2. Chinese Academy of Aerospace Systems Science and Engineering, Beijing 100037, China
3. Qi An Xin Technology Group Inc., Pangu Division, Beijing 100044, China
4. School of Cyberspace Security, Nankai University, Tianjin 300071, China

Abstract: To address the limited information coverage of single-feature static malware classification and the high cost of multi-feature fusion models, this paper proposes a lightweight collaborative method for resource-constrained scenarios. The method integrates opcode sequences and static API calls to characterize malware from instruction execution and system interaction perspectives. A multi-scale residual module is used in the opcode branch to capture local patterns and long-range dependencies, while a triplet semantic chain is introduced in the API branch to convert discrete calls into structured functional information, reducing sparsity and redundancy. A position-aware multi-head self-attention mechanism is then employed to dynamically fuse the two features for joint modeling, achieving high accuracy with low model co1mplexity. Experiments on the BIG2015 and enterprise-level PE datasets achieve accuracies of 99.53% and 99.4%, respectively, with only 30.2 MFLOPs, demonstrating a high-accuracy and low-overhead solution for malware classification in resource-constrained environments.

Keywords: malware classification, deep learning, lightweight, opcode, API

收稿日期：XXXX–XX–XX；**修回日期：**XXXX–XX–XX

通信作者：谢雪，xiexue2008@163.com；陈治国，chenzhiguo@nuist.edu.cn

基金项目：国家自然科学基金资助项目（No. 62102190）

Foundation Items：Project supported by the National Natural Science Foundation of China（No. 62102190）

0 引言

恶意软件是指未经用户授权、用于破坏系统或窃取数据的程序，包括病毒、蠕虫、特洛伊木马和勒索软件等，已成为当前网络安全领域最严峻的威胁之一[1]。伴随信息技术的快速发展，恶意软件的复杂性和多样性持续攀升，变种频繁且普遍具备规避传统检测方法的能力。根据 AV-TEST 最新数据，2025 年新增的恶意软件样本已超过 1 亿，周均增量达到百万级[2]。恶意软件分类作为恶意软件分析的核心环节，其目的是根据样本的行为特征或功能目的，将其精准划分至对应家族。这一过程能够揭示同一家族内部不同变体的共性行为模式与代码特征，为构建高效、准确的恶意软件检测引擎、自动化威胁情报生成以及攻击溯源与防御策略制定提供重要的基础支撑。然而，恶意软件的持续演化及其广泛采用代码混淆、加密和多态变形等技术，显著增加了分类难度，如何在高效性与准确性之间取得平衡，仍是当前研究的主要挑战。

传统恶意软件分类方法主要分为动态与静态分析两类。动态分析通过在沙箱中监控样本运行时的行为（如 API 调用序列、注册表修改及网络交互等）来捕捉恶意特征，在一定程度上能够应对代码加密、指令重排等混淆技术。然而，该方法需实际执行样本，导致较高的计算开销和安全风险[3]。此外，恶意软件常采用虚拟机检测、延迟触发等机制规避分析，其复杂的多路径触发行为亦难以全面覆盖。静态分析则无需执行代码，通过提取的二进制结构、操作码序列和字符串信息等静态特征进行识别，避免了动态分析的高资源消耗[4]。但传统依赖单一特征的静态方法存在明显局限：一方面，难以全方位捕捉恶意软件的多维度行为模式，易导致相似家族变种（如 Kelihos_ver1 与 Kelihos_ver3）之间的误分类[5]；另一方面，在面对代码混淆、加密和多态变形等规避技术时，其分类性能也显著受限[6]。

近年来，机器学习技术在恶意软件分类中得到广泛应用。相较于完全依赖人工规则的方法，机器学习模型能从大量数据中自动学习行为模式，通过学习样本的统计模式提升分类精度。传统机器学习方法主要依赖人工设计的特征，如 PE 文件结构、操作码序列、API 序列及字符串信息等，并结合支持向量机（SVM）、随机森林（RF）或朴素贝叶斯（NB）等分类器进行建模。尽管在特定场景下表现良好，但其特征工程高度依赖专家经验，难以全面覆盖恶意软件持续演化的多样化变种和对抗策略，导致模型在面对未知或变种样本时泛化能力不足，性能下降明显。

为克服传统机器学习方法对人工特征的高度依赖及其泛化能力不足的问题，基于深度学习的恶意软件分类方法逐渐成为研究热点。其中，基于静态分析的深度学习方法因无需执行样本即可高效处理大规模数据而备受关注[7]。该类方法能够直接从原始静态特征中自动学习生成判别力更强的特征表示，并利用深度神经网络实现端到端的建模与分类，显著增强了对混淆或变种样本的识别能力。

然而，现有基于静态分析的深度学习方法仍面临若干关键挑战：一方面，模型往往依赖单一静态特征进行建模，由于该类特征在语义覆盖范围上的局限性，难以充分刻画复杂且多样化的恶意行为模式，制约了模型的分类精度与泛化能力[8][9]；另一方面，为弥补单一特征表征能力不足，研究中逐渐引入多种静态特征进行融合，但不同特征在结构形式与语义表达上存在显著异构性，若缺乏有效的特征选择与融合机制，容易引入冗余信息，反而限制模型整体性能的进一步提升[10]。此外，多特征融合通常伴随着模型参数规模和计算复杂度的显著增加，在资源受限场景下难以兼顾实时检测与高效部署需求[11]。上述问题共同凸显了轻量级系统设计的必要性，即需要在统一框架下平衡特征表示能力、特征融合效果与计算效率[12]，以满足资源受限环境中对实时性和部署效率的实际要求。

针对上述挑战，本文选取操作码序列与静态 API 调用（通过分析反汇编后的 ASM 文件静态提取，而非运行时监测获得）作为核心特征，提出一种基于多特征融合的轻量级恶意软件分类方案。操作码序列从指令组织层面反映程序的底层结构特征，静态 API 调用则刻画程序对系统接口的潜在使用路径并且蕴含功能语义信息。实验结果表明，这两类特征从不同维度表征恶意软件的程序结构与功能特征，其协同建模能够有效提升模型的判别能力。本文的主要贡献如下：

1) 提出一种面向资源受限部署场景的轻量级静态恶意软件分类框架。与简单堆叠多种静态特征的做法不同，本文从系统设计角度出发，选取操作

码序列与静态 API 调用作为两类互补的核心特征源。操作码序列刻画程序在指令组织层面的结构执行特征，而静态 API 调用反映程序潜在的系统交互意图。通过在统一架构下对这两类特征进行协同建模，在有效提升恶意行为表征能力的同时，严格控制模型规模与计算开销，满足资源受限环境下的部署需求。

2) 设计一种兼顾语义压缩与多尺度结构建模的高效特征表示策略。针对静态 API 调用语义离散、冗余较高的问题，引入基于“动作–对象–类别”的语义三元组表示方法，将原始 API 调用压缩为结构化的功能语义信息；针对操作码序列中局部指令模式与长程结构依赖并存的特点，采用改进的一维 Res2Net 模块进行多尺度建模。该特征表示策略在统一的轻量化设计约束下，增强了语义表达能力与结构覆盖能力，而未引入额外的显著计算负担。

3) 提出一种具备位置感知能力的异构特征融合与分类机制。为有效融合操作码的结构特征与 API 的功能语义，本文设计了一种结合位置编码与多头自注意力机制的轻量级融合模块。该机制能够在保持序列结构信息的前提下，对来自不同特征源的关键成分进行自适应加权，从而提升融合特征的判别能力。同时，该模块在参数规模和计算复杂度上保持较低水平，适用于实时检测与资源受限部署场景。

4) 实现高精度分类性能：在 BIG2015 公开数据集上，本方法分类准确率达到 99.53%；在企业级 PE 数据集上，准确率达 99.40%，充分验证了方案在真实恶意软件分类场景下的适应性和有效性。

本文结构安排如下：第 1 节介绍了相关的恶意软件检测与分类方法；第 2 节详细阐述了本文提出的基于多特征融合的轻量级恶意软件分类方法；第 3 节展示并分析了实验结果；第 4 节总结全文并探讨未来的研究方向。

1　相关工作

1.1　基于操作码的恶意软件分析方法

操作码能够反映程序底层指令的逻辑结构，是静态恶意软件检测和分类中的核心行为特征之一。围绕操作码序列的建模方式，已有研究从序列表示、时序依赖建模以及多尺度特征提取等方面进行了探索。

Yuan 等人[13]针对深度学习模型对时间演进样本适应性不足的问题，提出了一种基于操作码序列的异常样本检测方法。通过识别并预筛选模型误判的异常样本，提高了模型对时间演化样本的适应能力。但该方法主要侧重于检测结果的后处理，对操作码特征本身表达能力的提升较为有限。为增强模型对复杂恶意行为的建模能力，Jeon 等人[14]将二进制文件解析为操作码片段集合，利用自编码器压缩操作码片段以缓解长期依赖问题，并结合动态递归神经网络进行分类，在实现 96% 准确率的同时增强了模型对变种样本的适应性。然而，该方法在处理如间接跳转等复杂控制结构时仍面临结构感知不足的局限。

考虑到操作码序列在不同长度、不同语义粒度下对建模效果影响显著，Sewak 等人[15]系统评估了 LSTM 网络在恶意软件分类中的超参数设置，分析了序列长度、嵌入维度与网络层数等参数之间的交互效应，并通过层次化调参策略将准确率提升至 99.21%。但结果表明，LSTM 在建模长操作码序列时存在性能瓶颈，难以高效捕捉复杂指令间的深层依赖关系。为克服序列建模的局限性并挖掘深层次的指令语义，Hao 等人[16]通过抽象操作数类型，融合 SIF、P2V 与 P-means 算法挖掘指令的结构与上下文语义信息，并引入通道注意力机制与空间金字塔池化模块增强特征表达能力，在 BIG2015 上实现了 99.40% 的分类准确率。该方法在复杂指令建模方面表现较好，但其复杂的特征融合结构限制了模型的可部署性。此外，针对混淆变种识别与类别不平衡的挑战，Darem 等人[17]提出融合图像转换与半监督深度学习的分类方法，其利用 PageRank 算法筛选高价值的 n-gram 操作码特征，并转换为灰度图像，增强模型对混淆变种的分类能力，在 BIG2015 数据集上取得了 99.12% 的分类精度。

1.2　基于静态 API 调用的恶意软件分析方法

静态 API 调用是通过解析反汇编代码获得的重要静态特征，能够从系统接口层面反映程序可能涉及的系统资源类型及其操作方式，为分析恶意软件的功能语义提供关键信息。

Gibert 等人[10]提出了一种融合字节序列、静态 API 调用和操作码序列的多特征融合方法，通过反汇编提取 API 调用并编码为二元向量，结合调用频

次共同构成语义特征表示。该方法一定程度上缓解了单一特征信息覆盖不足的问题，但特征冗余和过高的模型复杂度限制了模型的计算效率。为提升模型效率和对抗鲁棒性，Shaukat等人[18]设计了一种基于静态API调用与对抗训练的检测框架。该方法从PE文件中提取静态API调用特征，训练多个对抗神经网络模型以提高系统抵抗不同攻击策略的能力，将恶意样本的逃逸率降低至12%。尽管在鲁棒性方面表现优异，但该方法高度依赖于预定义攻击类型，泛化能力在未知攻击场景下仍需进一步验证。针对数据不平衡和多类别分类问题，Demirkıran等人[19]提出了Random Transformer Forest（RTF）集成模型，该模型首次将静态API调用与Transformer编码器结构结合，并引入BERT与CANINE预训练模型以丰富语义建模能力。通过Bagging策略集成多个Transformer分类器，有效提升了整体分类性能。然而，预训练模型的引入显著增加了训练开销，在资源受限的环境中难以部署。

1.3 基于多特征融合的恶意软件分析方法

传统静态恶意软件分类方法多依赖操作码、字符串、PE头或静态API调用等单一特征，难以适应恶意软件的多样化变种。为提升鲁棒性与泛化能力，研究者逐步转向多特征融合策略，通过联合建模不同维度的特征，从多个层面刻画恶意行为与结构特征。

Vinayakumar等人[20]通过融合操作码、API调用、系统调用及网络流量等多种静态特征，构建了基于卷积神经网络（CNN）与长短期记忆网络（LSTM）的混合模型，同时挖掘局部空间结构与时序依赖关系，在多类别恶意软件分类任务中实现了97.5%的分类准确率。然而，引入大量特征后模型结构复杂，训练成本较高。为减少计算开销，Huo等人[21]基于一维CNN对操作码序列与网络流数据进行联合建模，该方法通过卷积层提取关键行为特征，在保留特征表达能力的同时降低了模型复杂度，但对特征语义关联的建模仍不充分。为更深入理解特征间的语义结构，Do Xuan等人[22]将API调用序列构建为图结构，并融合操作码与系统调用特征，利用图神经网络（GNN）建模节点及其连接关系，有效增强了对复杂恶意样本的分类能力。Bensaoud等人[23]则采用N-gram与TF-IDF方案对操作码和API调用进行向量化编码，并利用CNN-LSTM架构实现序列建模，有效增强了模型的语义建模能力。针对静态分析场景下的稳健性问题，Xuan等人[24]将汇编指令与API调用联合建模，引入空洞空间金字塔池化（ASPP）结构以增强对多尺度行为特征的捕捉能力。并采用量子粒子群算法对模型超参数进行优化，有效提升了模型在混淆、指令插入等静态规避技术下的鲁棒性。

此外，随着Transformer架构在自然语言处理领域的成功应用，基于预训练语言模型的多特征融合方法也逐渐应用于恶意软件检测和分类领域。Babu等人[25]提取系统调用、文件操作及注册表修改等行为特征，借助Longformer结构建模长距离依赖，有效增强了对复杂恶意行为的建模能力。Xu等人[26]通过自监督学习训练出预训练模型MalBERT，专用于恶意软件的语义理解，在应对未知恶意样本时展现出较强的泛化能力。除预训练建模方法外，基于度量学习的Siamese网络也为恶意软件分析提供了新思路。该类方法通过共享参数结构学习样本间的相似性关系，能够在分布差异较大或样本不足的场景下提升特征表示的判别能力。基于此，Zhou等人[27]将GAN与Siamese结构结合，通过引入对抗生成机制缓解不同数据分布之间的差异，并利用相似性度量提升样本表示的区分能力。这种相似性学习的建模思想，同样可以应用在恶意软件分类中。例如，Hsiao等人[28]将恶意软件样本转换为图像表示，并结合Siamese网络开展分类研究，表现出较好的鲁棒性与分类效果，验证了该类方法在样本稀缺场景下进行恶意软件相似性判别的可行性。

综上所述，多特征融合方法通过整合操作码、静态API调用、字符串、PE结构等静态特征，以及系统调用、网络流量等动态特征，可以有效增强恶意软件的特征表征能力；同时，预训练模型与Siamese network等方法的引入，也进一步拓展了恶意软件分析在语义建模和相似性学习方面的研究思路。但在高维特征融合场景下，现有深度模型普遍面临特征冗余和计算开销较高的问题，限制了其在实时和资源受限场景中的实际应用。

2 所提方法

2.1 方法概括

本文面向资源受限与工程可部署场景，优先选

取无需执行样本即可稳定获取、便于批量处理的静态特征，以避免动态监测对运行环境、触发条件及反沙箱对抗机制的强依赖。因此，本文以反汇编生成的ASM文件为统一入口提取两类核心输入：操作码序列与静态API调用。理论上，这两类特征在信息维度上具有互补性：操作码序列刻画指令级执行结构与控制流模式，静态API序列刻画程序与系统资源交互的功能意图。基于该互补性进行联合建模，可在较低模型规模下获得稳定的性能增益。第3.3节的消融实验也表明，融合两类特征更有利于提升模型对不同家族的区分度。所提出的恶意软件分类系统架构如图1所示，主要由信息收集、特征处理、特征融合和分类器构建组成。

1) 信息收集部分：从恶意软件反汇编生成的ASM文件中提取关键静态特征，包括操作码序列和静态API调用序列，为后续特征建模与分析提供原始序列输入。

2) 特征处理部分：针对不同类型静态特征的结构与语义特点，分别设计对应的特征建模方法。对于操作码序列，引入改进的Res2Net模型进行多尺度特征提取，从底层指令序列中捕获局部指令模式与长程结构依赖关系；对于静态API调用序列，构建基于三元组的语义链表示，以结构化刻画程序潜在的功能特征。

3) 特征融合部分：该部分首先将API分支提取的功能语义特征与操作码分支获得的多尺度特征在特征维度上进行拼接，随后引入位置编码补充序列位置信息，并结合缩放点积注意力机制刻画特征间的自相关关系，自动关注对恶意行为判定贡献较高的关键特征组合，实现跨模态特征的动态加权与深度聚合。

4) 分类器构建：分类器采用多层感知机结构完成特征映射与分类判别。融合特征经维度转置（Permute）和全局平均池化压缩后，输入由Linear、ReLU和Dropout组成的全连接网络。输出层给出各恶意软件类别的概率分布，实现准确分类。

第2.2节至第2.5节将分别阐述上述部分的具体设计与实现。其中，特征处理模块的具体内容将在第2.3节和第2.4节分别介绍。

2.2　信息收集

2.2.1　操作码序列提取

为从数据集的ASM汇编源文件中提取高质量操作码序列，本文采用基于正则表达式的预处理策略。具体而言，利用正则模式\s([a-fA-F0-9]{2}\s)+\s*([a-z]+)精确匹配并截取汇编行中的操作码字段。同时，为确保特征序列的纯净度与行为表征的准确性，本文剔除了包括debug、line、macro、endm在内的调试伪指令及其他冗余噪声。

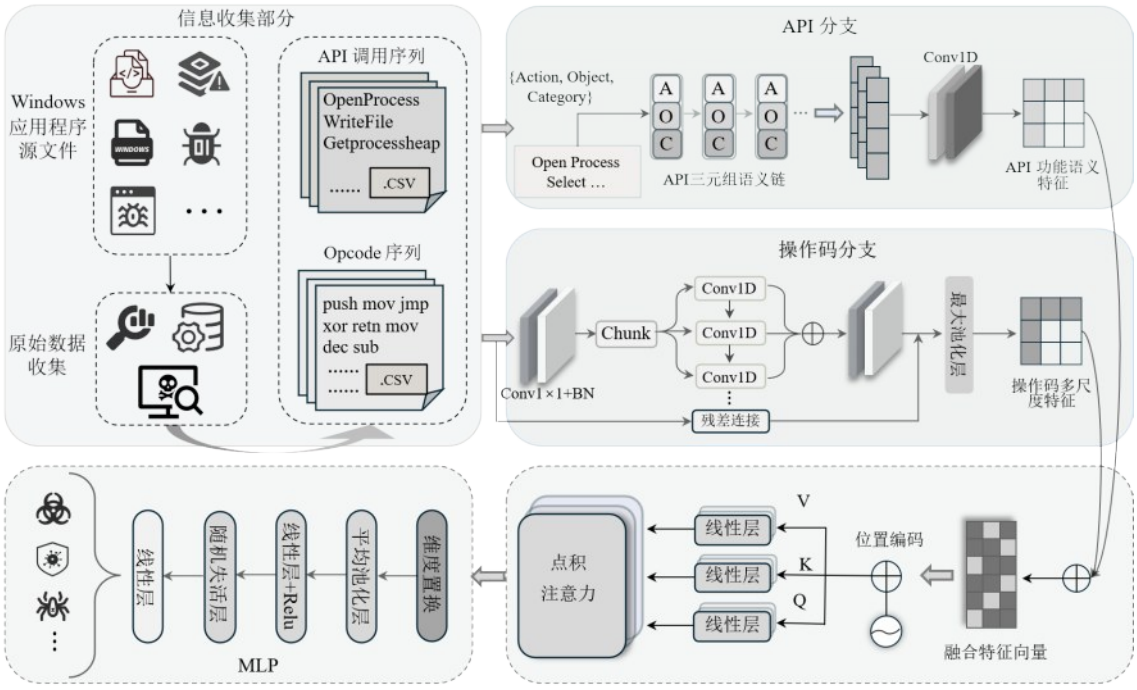


图1　恶意软件分类系统架构

为进一步优化序列表达、去除冗余并增强判别能力，对提取到的操作码序列进行连续重复操作码压缩处理。例如，原始序列{*a, a, b, c, c, c, d, a, a, ...*}经处理后变为{*a, b, c, d, a, ...*}，该处理能够有效缓解由循环结构和编译器特性引入的冗余，同时保留对后续建模至关重要的指令模式信息。

2.2.2 静态API调用序列提取

静态API调用序列通过对反汇编生成的ASM文件进行结构化解析获得，利用正则表达式识别具有明确语义的函数调用指令，在无需执行恶意样本的情况下构建API调用序列。恶意程序中常因循环结构、异常处理或混淆策略而产生大量重复API调用。为减轻此类冗余对特征建模的影响，本文对连续重复出现的API调用进行合并，仅保留一次出现，降低冗余噪声对特征建模的干扰。

2.3 操作码特征处理

操作码建模方法主要有两类：一类是将操作码序列视为离散Token，利用序列模型进行上下文语义学习。该类方法实现较为方便，但操作码序列通常较长，容易带来较高的计算开销。另一类是结合操作码之间的执行关系或控制转移关系构造成图进行语义学习，如指令依赖图、控制流图等。该类方法结构表达能力较强，但通常依赖额外的结构构建过程，工程复杂度和推理成本相对较高。

2.3.1 操作码序列嵌入

为将操作码序列转化为神经网络可处理的向量表示，基于处理后的序列构建由324个唯一操作码组成的嵌入词表。每个操作码通过嵌入层被映射为k维密集向量表示。设$\boldsymbol{f}_i \in R^k$表示第i个操作码的k维嵌入向量，则长度为n的操作码序列表示为

$$\boldsymbol{F} = [\boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{f}_3, ..., \boldsymbol{f}_n] \in R^{(n \times k)} \tag{1}$$

其中，n为操作码序列长度，k为嵌入维度。矩阵$\boldsymbol{F}$即为操作码序列的密集特征表示。

2.3.2 操作码多尺度特征提取

传统卷积神经网络处理序列数据时多采用固定大小的卷积核，虽然能够有效捕获局部指令模式，但在建模跨越较长指令跨度的结构依赖关系方面能力有限。在恶意软件分析中，单个操作码仅对应底层操作（如参数准备、内存访问），而恶意行为往往由多个局部模式在较长指令序列中组合形成。因此，单尺度卷积特征难以同时兼顾操作码序列的局部特性与整体结构特征。

为应对这一问题，本研究在对比几种经典的多尺度模块后引入Res2Net模块，与传统多尺度结构相比，Res2Net通过通道分组和逐级残差融合实现不同粒度特征的高效建模。考虑到原始Res2Net主要针对二维数据设计，本文将其中的二维卷积替换为一维卷积，以更好地适配操作码序列的线性结构和序列依赖特性。

如图2所示，将操作码特征$\boldsymbol{F}_{(1:n)} \in R^{(n \times k)}$沿通道维度划分为$n$个子尺度特征$\{\boldsymbol{X}_0, \boldsymbol{X}_1, \cdots, \boldsymbol{X}_n\}$，$\boldsymbol{X}_i \in R^{n \times k_i}$表示第$i$个尺度的特征，$\boldsymbol{k}_i$是该尺度下的特征分量。对每个子组特征$\boldsymbol{X}_i$通过一维卷积提取局部指令模式，并通过逐级残差融合在不同尺度之间聚合上下文信息[29]。

$$\boldsymbol{Y}_i = Conv(\boldsymbol{X}_i) + \boldsymbol{Y}_{i-1}, i = 1, ..., n \tag{2}$$

其中，$\boldsymbol{Y}_i$为第i个尺度的输出特征。第一个尺度$\boldsymbol{Y}_0 = \boldsymbol{X}_0$作为初始信息直接参与特征融合，保留原始操作码特征。经过卷积和残差融合后的操作码特征$\boldsymbol{Y}_0, \boldsymbol{Y}_1, \cdots, \boldsymbol{Y}_n$积累了不同尺度的信息，将其在通道维度上进行拼接，并通过逐点卷积映射回原始特征维度，得到输出特征$\boldsymbol{Y}_{out}$。

$$\boldsymbol{Y}_{out} = Conv_{1 \times 1}(Concat(\boldsymbol{Y}_0, \boldsymbol{Y}_1, ..., \boldsymbol{Y}_n)) \tag{3}$$

为保证信息传递的稳定性并增强非线性表达能力，在输出$\boldsymbol{Y}_{out}$和初始输入$\boldsymbol{F}_{1:n}$间加入残差连接与归一化操作，得到最终的操作码特征表示$\boldsymbol{F}_{out}$。

$$\boldsymbol{F}_{out} = RELU(BN(\boldsymbol{Y}_{out}) + \boldsymbol{F}_{1:n}) \tag{4}$$

与Inception1D、MobileNetV2等多尺度建模方法相比，改进后的Res2Net模块通过通道分组与并行卷积显著降低了计算开销，其逐级残差融合机制尤其适用于包含局部模式和长程结构依赖的操作码指令序列，能够有效提升操作码特征对恶意软件多样化行为模式的表征能力。

2.4 静态API调用特征处理

现有基于API的语义表示大致可分为两类：一类将API调用序列视为词序列，通过词向量/序列模型（如Transformer）学习上下文语义；另一类构建API调用图或控制流图，借助图神经网络建模结构依赖。前者通常需要较大的词表与嵌入空间来覆盖大量离散API名称，容易造成高维特征稀疏与冗余并存，后者额外的图构建学习带来较高的模型规模与推理开销，不利于资源受限场景部署。

为兼顾语义表达与轻量化需求，本文采用结构

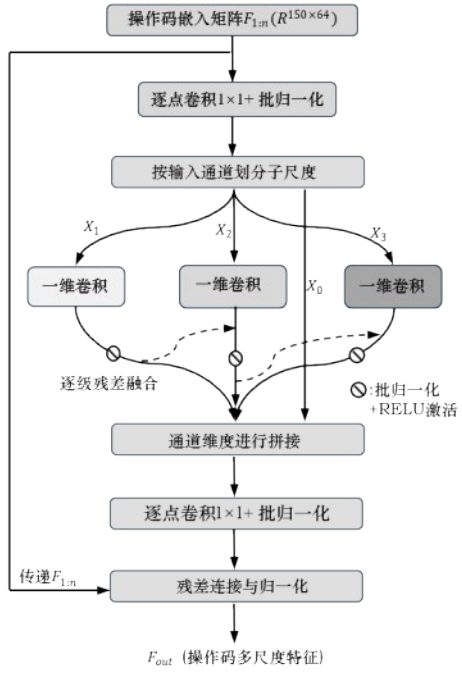


图2　操作码多尺度特征提取模块

化语义抽象策略而非单纯统计词袋或大规模序列语义模型：将离散 API 调用映射为(Action, Object, Category)的三元组语义单元，用更稳定的语义单元替代原始离散API调用名，在实现语义压缩、降低特征冗余的同时保留关键语义信息。

2.4.1　API 三元组语义链构建

通过分析 Windows 系统API的命名规范，发现API名称通常由表示操作行为的动词（Action）和表示对象的名词（Object）组成，蕴含丰富的语义信息。

基于此，本文设计了一种三元组表示方法，将离散的API调用转换为结构化的（动作、对象、类别）的结构化特征，并根据API对系统资源的实际影响程度将其划分为三类[30]：Select类API主要用于查询操作，包括读取系统信息、获取资源状态等；Update类API涉及对系统资源的修改，如创建、删除或写入操作；Other类则包含调试相关或难以明确归类的API调用。表1列举了较为常见的API及其动作标签与类别标签的映射示例，以揭示API调用对系统资源的影响程度。

本文首先通过正则化分词和语义匹配，从API名称中抽取核心动作a_i与操作对象o_i，并根据预定义的类别映射表为其分配功能类标签c_i。从而将API调用转化为语义三元组t_i=(a_i, o_i, c_i)。其中，动

表1　　API与动作标签映射示例

API函数名	映射动作	类别标签
CreateFileA	create	Update
DeleteFileA	delete	Update
OpenProcess	open	Select
VirtualAllocEx	allocate	Update
IsDebuggerPresent	is	Other
LoadLibraryA	load	Update
TerminateProcess	terminate	Update

作项a_i通过预先构建的动作词表进行匹配：该词表由从常见API中提取，共包含70个核心动作，包含了API调用的核心操作。随后，依据动作词将其映射到三类粗粒度功能类别{Select, Update, Other}，以提升语义抽象的稳定性与泛化性。操作对象o_i则由从API名称中剥离动作词后得到的剩余字符串构成。

基于上述解析规则，本文将动作词、对象词与类别标签统一编码为可学习Token，并构建总Token词表规模为2499（包含Action/Object/Class及占位符），用于后续张量化与模型训练。为确保特征提取的完备性与一致性，针对无法命中动作词规则的API调用，本文将其对象字段统一归一化标记为Object，并将类别字段定义为Other，以保证语义链在张量化过程中的可用性与维度一致。通过脚本统计，此类缺乏显式动作词的API调用共2952个，在数据集中占比不足1%，因此该归一化策略不会在整体上造成明显的语义缺失。经上述处理，原始API调用序列被转化为结构化的三元组语义链$T=\{t_0,t_1,\cdots,t_n\}$。如表2所示，恶意行为（如进程注入、反调试）通常伴随特定语义的API调用，表中列举了相关典型API及其对应的三元组表示。

2.4.2　API 功能语义信息提取

在完成语义链构建的基础上，本文通过建立全局统一的语义词典，将“动作、对象、类别”三类异构语义单元映射至同一特征空间，为后续的特征学习提供通用的编码基础。

1) 嵌入映射：词典的大小为V，其中每个词元均对应一个k维向量，由此构建嵌入矩阵$\boldsymbol{W}=R^{(V\times k)}$，基于该矩阵，将三元组$t_i=[a_i,o_i,c_i]$中的各元素转化为向量并拼接，得到$k$维向量$\boldsymbol{E}_i=[\boldsymbol{e}_{a_i},\boldsymbol{e}_{o_i},\boldsymbol{e}_{c_i}]$。故长度为$n$的API三元组语义链$T=$

$\{t_0,t_1,\cdots,t_n\}$ 可转化为对应的嵌入向量序列 $\{\boldsymbol{E}_0,\boldsymbol{E}_1,\cdots,\boldsymbol{E}_n\}$。

2) 功能语义特征提取：为挖掘语义链中的功能语义，本文采用步长为3的一维卷积对其进行局部特征提取，确保卷积窗口在滑动时覆盖一个完整的三元组单元，从而实现三元组内部动作、对象与类别向量的特征聚合。卷积运算如公式(5)所示。

$$\boldsymbol{Z}_i = \sigma(W_{conv} \bullet \boldsymbol{E}_i + b_{conv}) \tag{5}$$

其中，σ为*sigmoid*激活函数，W_{conv}为卷积核参数（其感受野大小覆盖一个三元组），b_{conv}为偏置项，$\boldsymbol{Z}_i$为最终的API语义特征表示。通过上述处理，原始离散的API调用被转换为包含深层功能语义信息的低维稠密特征表示。

2.5　特征融合与分类器构建

在实际的恶意软件行为分析中，单一特征难以完整刻画复杂攻击模式。操作码序列能够从微观层面反映底层指令的结构逻辑，而静态API调用则从宏观交互层面揭示功能意图与系统影响。为说明操作码序列与静态API调用之间的互补关系，本文以蠕虫类恶意软件中常见的远程进程注入行为为例进行分析[31]。如图3所示，反汇编后的操作码序列展现了与进程注入相关的指令模式。诸如PUSH offset 'target.exe'之类的指令为随后的函数调用构建参数，CALL OpenProcess对应进程句柄的获取操作；随后出现的内存分配和写入相关的指令块，反映了远程内存操作的实现过程；同时，CreateRemoteThread的调用体现了远程线程创建功能在静态代码层面的组织方式。该操作码序列从指令组织层面刻画了蠕虫类恶意代码中进程访问与代码注入功能的静态实现结构。

与之对应，API调用从功能语义层面补充了上述行为模式。例如，OpenProcess表明程序具备访问目标进程的能力，VirtualAllocEx和WriteProcessMemory描述了远程内存分配与写入行为，而CreateRemoteThread则反映了触发远程执行的能力；其他API（如LoadLibraryA）进一步提供了与运行时环境准备相关的上下文信息。这些API在反汇编代码中的结构化共现，构成了基于进程注入的恶意软件的典型功能语义模式。

在本文中，上述API被映射为由“动作－对象－功能类别”组成的语义三元组，从而在不依赖动态执行路径的前提下，对恶意意图进行结构化刻画。不同类型的恶意软件（如蠕虫、木马和勒索软件）在操作码序列与静态API调用特征上通常呈现出差异化侧重，这种差异不仅体现在局部特征形态

表2　API三元组语义链的示例片段

API调用	API功能概述	对应的语义三元组 (t_i)
OpenProcess	打开目标进程并获取句柄，为后续代码注入做好准备。	(open, Process, Select)
ReadProcessMemory	允许读取指定进程的内存，用于窃取或注入敏感数据。	(read, ProcessMemory, Select)
VirtualAllocEx	在目标进程的地址空间内分配内存，以存储恶意代码。	(allocate, Object, Update)
WriteProcessMemory	向目标进程写入数据或代码片段。	(write, ProcessMemory, Update)
CreateRemoteThread	在目标进程中创建线程以执行注入的代码。	(create, RemoteThread, Update)
IsDebuggerPresent	确定调用进程是否正在调试，用于反调试。	(is, DebuggerPresent, Other)

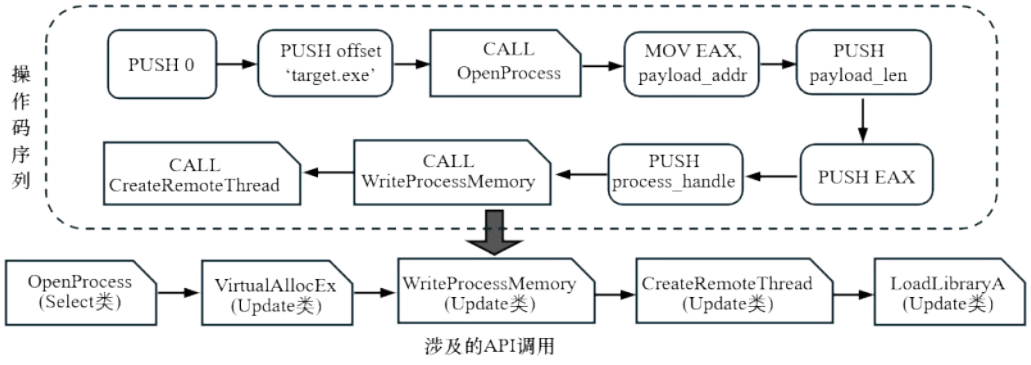


图3　蠕虫远程进程注入行为示例

上，也反映在跨特征维度的整体结构关联关系中。仅依赖简单特征拼接难以充分刻画操作码结构特征与API功能语义之间的内在关联。为此，本文在分类层之前引入位置感知的多头自注意力机制。如图4所示，多头自注意力机制用于对融合特征中不同位置的要素进行自适应加权，以捕获特征内部及跨特征维度的依赖关系。但MHSA在计算注意力权重时仅依赖查询（Query）与键（Key）的内容相似度，未显式编码特征在序列表示中的位置信息[32]，在处理具有结构顺序特性的静态恶意软件特征时，可能导致部分结构信息被弱化。针对该问题，本文在多头自注意力机制之前引入位置编码层，通过公式(6)将位置结构信息嵌入融合特征向量。

$$\begin{aligned} PE_{(pos,2i)} &= \sin\left(pos/10000^{2i/d}\right) \\ PE_{(pos,2i+1)} &= \cos\left(pos/10000^{2i/d}\right) \end{aligned} \tag{6}$$

其中，pos表示Token在序列中的位置，i表示位置编码的维度，d为嵌入向量的维度。经过位置增强的特征随后被送入多头自注意力机制进行加权建模。即通过三个独立的线性变换层分别生成查询矩阵$\boldsymbol{Q}$、键矩阵$\boldsymbol{K}$和值矩阵$\boldsymbol{V}$：$\boldsymbol{Q}_i = \boldsymbol{F}\boldsymbol{W}_i^Q, \boldsymbol{K}_i = \boldsymbol{F}\boldsymbol{W}_i^K, \boldsymbol{V}_i = \boldsymbol{F}\boldsymbol{W}_i^V$。其中，$F$是融合特征向量，$\boldsymbol{W}_i^Q$、$\boldsymbol{W}_i^K$、$\boldsymbol{W}_i^V$为对应的可学习权重矩阵。这些矩阵被投影到不同的注意力子空间，在每个子空间内按照公式(7)计算注意力权重。

$$Attention\left(\boldsymbol{Q}_i,\boldsymbol{K}_i,\boldsymbol{V}_i\right) = SoftMax\left(\frac{\boldsymbol{Q}_i\boldsymbol{k}_i^T}{\sqrt{\boldsymbol{K}_i}}\right)\boldsymbol{V}_i \tag{7}$$

各子空间的输出特征通过特征维度的级联操作进行整合，并经由可学习的线性变换矩阵$\boldsymbol{W}_0$融合生成最终的增强特征表示：$MultiHead\left(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\right) = Concat\left(head_1,\cdots,head_h\right)\boldsymbol{W}_0$，经池化降维并输入至由全连接层构成的分类器中完成分类。

3　实验与结果分析

为系统评估所提出方法的有效性与实用性，本文从分类性能、模型结构贡献、计算效率以及跨场景泛化能力等多个维度开展了全面实验分析。实验分别基于BIG2015公开数据集与企业级PE恶意软件数据集进行，覆盖基准测试与真实应用场景验证两类需求。

在模型训练配置方面，统一采用PyTorch框架进行实现，批量大小设为128，优化器选用AdamW，损失函数为交叉熵损失。所有对比实验均在相同硬件与训练配置下完成，以确保结果的公平性与可复现性。整体实验设计分为以下三个递进式阶段：

1) 第一阶段侧重于融合特征的有效性验证。重点评估操作码序列与静态API调用融合特征的判别能力。在第3.2节中，分别将融合特征应用于传统机器学习模型和主流深度学习序列模型，通过分类性能、计算复杂度与推理效率等指标，验证多特征协同建模策略的有效性。

2) 第二阶段通过消融实验与结构对比实验分析模型各组成模块的实际作用。在第3.3节中，基于BIG2015数据集定量评估操作码分支、API分支以及位置编码与多头自注意力机制对整体性能的贡献；在第3.4节中，进一步通过超参数优化与结构设计实验，从参数效率与结构合理性角度验证模型

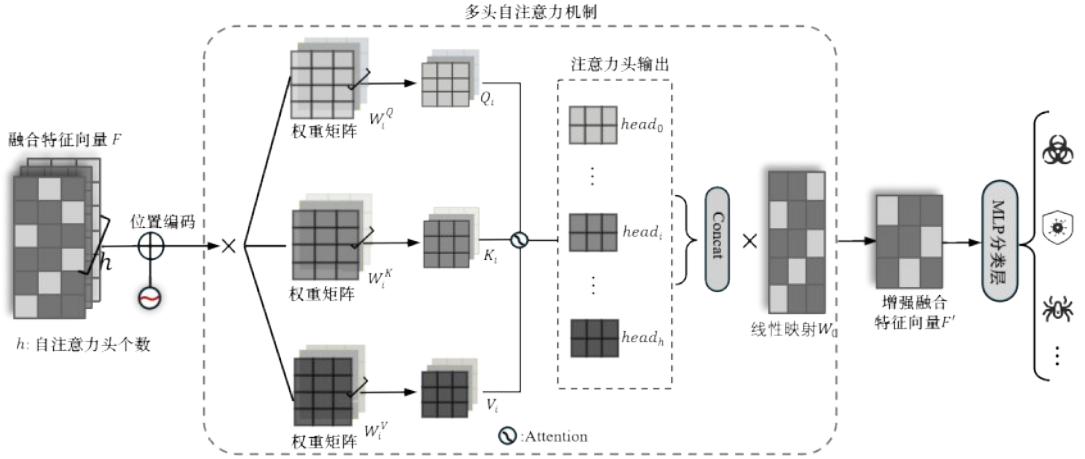


图4　特征融合与分类器构建

设计的有效性与稳定性。

3) 第三阶段旨在综合对比与评估泛化能力。在第3.5节中，从模型规模、推理速度和分类性能等指标出发，将本文方法与近年来具有代表性的研究工作进行系统对比；同时引入企业级PE数据集，对模型在真实应用环境下的鲁棒性与实用性进行验证；并通过跨数据集实验，评估模型在未知恶意软件家族场景下的泛化能力。

3.1 数据集

本文选取BIG2015公开数据集与企业级PE恶意软件数据集作为实验数据来源。前者作为主流研究对比的统一基准；后者用于验证模型在复杂真实环境中的泛化能力与工程实用性。

3.1.1 BIG2015数据集

BIG2015是由微软研究院与Kaggle于2015年联合发布的权威恶意软件分类基准数据集，广泛应用于静态恶意软件分析研究[33]。该数据集包含9个面向Windows平台的恶意软件家族，共计21,741个样本。其中，训练集包含10,868个带有家族标签的样本，测试集包含10,873个未标注样本。每个样本提供.bytes和ASM两种格式的原始文件。本研究选取其训练集的10,868个恶意样本，通过无放回采样按照7:3的比例随机划分为训练集和测试集，用于评估分类模型的性能。其样本具体分布如表3所示。

表3 BIG2015数据集的家族及其数量

恶意软件家族	数量/个	恶意软件家族	数量/个
Ramnit	1541	Tracur	751
Lollipop	2478	Kelihos_ver1	398
Kelihos_ver3	2942	Obfuscator.ACY	1228
Vundo	475	Gatak	1013
Simda	42	总计	10868

3.1.2 企业级PE数据集

为验证所提方法在真实工业环境中的适用性与泛化能力，本文联合信息安全企业构建了一个企业级PE恶意软件数据集。该数据集来源于企业日常安全运营与持续威胁监测系统，能够较为真实地反映实际恶意软件分类场景中的威胁分布特征。样本的家族标签由企业内部自动化分析系统与人工安全分析相结合确定。此外，数据集样本覆盖多个时间阶段，包含不同时期活跃的恶意软件家族及其变种。实验中，本文将所有PE样本统一反汇编为ASM文件进行特征提取，具体分布如表4所示。

表4 企业级PE数据集的家族及其数量

恶意软件家族	数量/个	恶意软件家族	数量/个
Autoit	1633	Msil	763
Barys	928	Nitol	1018
Emotet	380	Sivis	2372
Fragtor	312	Stormattack	3763
Fsysna	2163	Trickbot	153
总计	13485	-	-

3.2 特征有效性验证实验

3.2.1 基于机器学习的基线模型对比研究

为验证基于操作码与静态API调用的融合特征在恶意软件分类中的有效性，实验选取K-近邻（KNN）、决策树（DT）、随机森林（RF）和极限梯度提升（XGBoost）四种典型机器学习算法作为基线模型进行对比分析。

实验结果如表5所示，基于融合特征构建的基线模型均取得了较为理想的分类性能。其中，XGBoost在准确率和F1值两项核心指标上均超过95%，显著优于KNN与DT模型，表明操作码与静态API特征在融合后能够形成具有较强判别能力的特征空间。

表5 机器学习基线模型的特征效果验证

模型	分类准确率	F1值
KNN	86.16%	85.54%
DT	85.83%	85.79%
RF	90.21%	90.18%
XGBoost	95.82%	95.73%
本文模型	**99.53%**	**99.53%**

3.2.2 基于深度学习的序列模型对比研究

在验证融合特征有效性的基础上，为进一步评估本文模型在序列建模层面的性能优势，本研究将所提出方法与三种具有代表性的深度学习序列模型BiGRU[34]、BiLSTM[35]和DeBERTa[36]进行对比。这三类模型分别代表了循环神经网络、门控循环结构以及基于Transformer的自注意力架构，是当前恶意软件分析领域的主流技术路线。

在模型配置方面，为控制参数规模与对比的公平性，BiGRU与BiLSTM均采用单层结构，隐藏层维度统一设为128；DeBERTa模型设置为4层Transformer编码器，每层包含4个注意力头。除分类性能指标外，实验同时引入模型参数量（Params）与浮点运算量（FLOPs）作为计算复杂度衡量标准，为实际部署提供参考依据。

实验结果如表6所示，三种对比模型在BIG2015数据集上均取得了较高的整体分类性能。其中，BiGRU和BiLSTM在参数规模和训练速度方面具有一定优势，并且整体分类性能较高，但在处理样本数量较少的恶意软件家族（如Simda）时表现明显下降。这表明，基于循环结构的模型在长尾分布和数据稀缺场景下泛化能力有限，难以稳定刻画复杂行为模式。而DeBERTa依托自注意力机制在整体分类性能上优于BiGRU和BiLSTM，对

表6　融合特征在深度学习序列模型上的效果验证

模型	训练时间	参数量/KB	FLOPs/M	分类准确率	F1值
BiGRU[34]	109s	**403.5**	**16.2**	98.35%	98.28%
BiLSTM[35]	**104s**	459.2	20.4	98.63%	98.60%
DeBERTa[36]	255s	11099.6	101.1	99.13%	99.12%
本文模型	124s	533.5	30.2	**99.53%**	**99.53%**

Simda家族的识别准确率提升至83%，但其模型规模和计算开销显著增加，FLOPs达到101.1MFLOPs，在实时检测和资源受限环境中部署成本较高。

在NVIDIA RTX 4090平台上的测试结果表明，本文所提出模型在保持533.5KB参数规模的同时，实现了99.53%的分类准确率和99.53%的F1值，整体计算复杂度为30.2 MFLOPs，单样本平均推理时延仅为0.12ms。与DeBERTa相比，计算开销显著降低。这表明，本文模型在轻量化设计与高精度分类之间实现了良好平衡，更适用于实际网络安全场景中的在线检测与快速响应需求。

3.3　消融实验研究

为进一步分析模型各组成模块对整体性能的贡献，本节通过系统性的消融实验对操作码分支、API分支以及多头自注意力机制与位置编码的协同作用进行验证。实验保持统一的训练配置与超参数设置，确保实验结果具有可比性。

1) 特征提取模块验证实验

在特征层面，本文重点评估多特征融合策略相对于单一特征建模的性能提升效果。实验分别构建仅使用操作码特征、仅使用静态API特征以及融合两类特征的模型进行对比，实验结果如表7所示。可以观察到，仅使用操作码特征的模型在分类准确率上达到98.29%，表现优于仅使用API特征的模型（97.57%），说明操作码特征在刻画程序行为模式和指令结构方面具有更强的判别能力。当两类特征进行融合后，模型分类准确率和F1值进一步提升至99.53%，较单操作码模型和单API模型分别有不同程度的上升。该性能提升主要源于操作码特征与静态API特征在表征层面的互补关系，操作码特征侧重于描述程序在指令级别的执行结构与控制流特性，而API特征则从系统调用与资源交互角度抽象程序的功能意图，二者的联合建模能够构建更加完整、稳定的特征表示。

此外，融合特征模型在显著提升分类性能的同时，仍保持了优异的计算效率。其参数量控制在533.5KB，浮点运算量仅为30.2 MFLOPs，能够满足实际部署场景对模型规模和推理效率的要求。

2) 特征融合模块验证实验

本文首先将跨特征建模明确界定为：在融合阶段，模型能够通过显式的权重分配机制自适应建立操作码结构表征与API语义表征之间的依赖关系，实现选择性信息交互，而非仅进行特征并置。在此定义下，本节围绕协同建模的关键实现环节开展消融实验，分析融合阶段不同结构设计对分类性能的影响，从而验证该模块是否带来跨特征关联学习与性能增益。相关实验结果如表8所示。

表7　不同特征组合对模型性能的影响

模型	训练时间	参数量/KB	FLOPs/M	分类准确率	F1值
仅API分支	86s	325.9	27.8	97.57%	97.49%
仅操作码分支	**97s**	**225.5**	**26.4**	98.29%	98.27%
本文模型	124s	533.5	30.2	**99.53%**	**99.53%**

采用多头自注意力机制（MHSA）和位置编码（PE）的完整模型在准确率和F1值两个指标上均达到最优表现，当同时移除MHSA和PE时，虽然模型的参数量降至236.5KB，但分类准确率也下降了约1.2%。该结果表明，虽然去除融合模块能够降

表8　位置编码和多头自注意力机制对模型性能的影响

模型	训练时间	参数量/KB	FLOPs/M	分类准确率	F1值
无PE	118s	533.5	30.2	99.00%	99.00%
无MHSA与PE	**85s**	**236.5**	**3.57**	98.35%	98.25%
本文模型	124s	533.5	30.2	**99.53%**	**99.53%**

低计算成本，但会显著削弱模型对融合特征中关键判别信息的建模能力。在仅移除PE时可以发现，模型在参数量、训练时间和FLOPs等指标上基本保持不变，但分类准确率仍下降约0.53%。说明位置编码在特征融合阶段并非仅作为附加信息存在，而是有效刻画了融合特征之间的位置与结构关系。

为进一步验证所提出的融合模块在推理过程中是否产生跨特征协同，本文给出MHSA的跨模态注意力可视化结果（见图5）。具体而言，在融合层计算时，我们将操作码分支的Token表示作为Query，将API语义分支的Token表示作为Key，从而得到“在更新每个操作码位置的融合表示时，其对不同API语义位置的关注权重”的分布。图中高亮区域表示该操作码位置在融合过程中对特定API位置分配了更高的注意力权重。

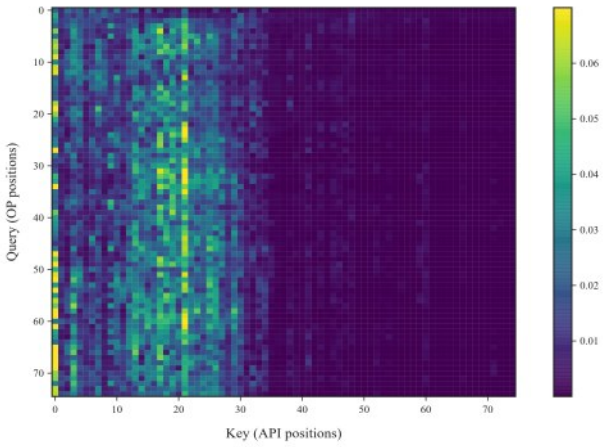


图5　跨模态注意力可视化结果

从图5的高亮区域可以观察到，模型会选择性地聚焦API语义链中的若干关键片段，并与操作码的结构特征进行融合学习。这说明两类异构特征之间存在显式的信息交互通道，且交互强度由注意力权重进行自适应调节，同时也表明MHSA在判定恶意样本时会建立操作码结构与API语义片段之间的显式关联，从而为前述“MHSA+PE提升融合判别能力”的结论提供可视化佐证。

此外，为排除性能提升仅来自参数规模增加的可能性，本文补充进行了一组近似同参数量级的对比实验：在保持两条特征提取分支与分类头完全一致的前提下，将融合阶段的MHSA与PE替换为简单拼接+两层MLP，并通过调整MLP隐藏层维度使两者参数量几乎一致（MHSA：2083.94KB，Concat+MLP：2082.44KB）。实验结果表明，在参数量基本相同的条件下，MHSA仍取得更优性能（Acc=99.53%，F1=99.29%），而Concat+MLP为Acc=99.10%，F1=99.00%。该结果说明，融合阶段的性能增益主要来源于注意力机制对跨特征依赖关系的建模能力，而非模型参数的简单扩张。

3) 数据预处理效果验证实验

为验证数据预处理阶段“连续重复项压缩”策略的合理性，本文进一步开展了消融对比实验。该策略将操作码序列与API调用序列中的连续重复项合并为单次出现，以减少由循环、填充与冗余调用带来的序列冗余。本文在相同的训练配置、模型结构与数据划分下，对比压缩前与压缩后的分类性能。训练过程如图6所示，两种设置均能稳定收敛，且压缩后的曲线整体略优于压缩前。在测试集上，压缩后模型的Accuracy和Recall分别为99.51%、99.50%，而未压缩时分别为99.31%、99.31%。这意味着该预处理并未导致识别能力下降，反而带来了轻微提升。

需要说明的是，在本文任务与特征表示方式下，模型学习并不依赖于连续重复次数这一显式频次信息。因此，连续重复项压缩可作为一种有效且稳健的预处理手段，在降低冗余与开销的同时保持（甚至提升）分类性能。

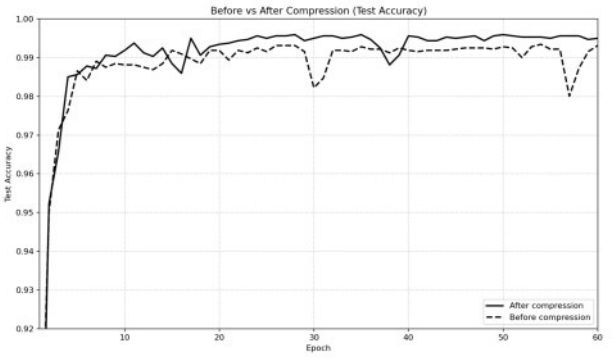


图6　序列压缩前/后的训练过程对比

3.4　模型结构验证实验

3.4.1　超参数优化分析

为确保模型在分类性能与计算效率之间达到最优平衡，本文采用网格搜索对关键超参数进行系统性调优。如表 9 所示，超参数搜索空间覆盖模型中对性能影响较为显著的多个维度，具体包括：操作码和 API 的嵌入维度、API 语义链的卷积核尺寸、Res2Net 模块的尺度分组数量、多头自注意力机制的头数、多层感知机中的 Dropout 率以及模型学习率。

表 9　　超参数搜索空间与最优值

超参数	搜索空间	最优值
操作码嵌入维度	{256, 128, 64}	64
API 嵌入维度	{256, 128, 64}	64
API 语义链卷积核大小	{2, 3, 4, 5}	3
改进后 Res2Net 分组数	{3, 4, 5, 6}	4
自注意力头数	{2, 3, 4, 5, 6}	4
Dropout 率	{0.2, 0.3, 0.4,0.5}	0.2
学习率	$\{10^{-4}, 10^{-3}, 10^{-2}\}$	10^{-2}

在超参数搜索过程中，所有实验均在相同的数据划分与训练配置下进行，并在训练阶段引入早停机制。从训练集中划出四分之一作为验证集，根据验证集损失的变化动态终止训练，有效抑制过拟合现象，确保所获超参数组合的泛化能力。

3.4.2　操作码分支多尺度模块对比实验

为进一步验证改进 Res2Net 模块的高效性，本文选取了多种具有代表性的轻量级多尺度卷积结构进行对比实验，分别包括时序卷积网络 TCN[37]、Inception1D[38]、单层空洞卷积 DilatedConv[39] 和 MobileNetV2[40]，这些模块均具备不同形式的多尺度感受野设计。

实验结果如表 10 所示，改进后的 Res2Net 模块在计算量与分类性能间实现了最佳平衡，以较低的计算成本获得了 99.53% 的分类准确率和 F1 值，整体表现优于对比方法。深入来说，DilatedConv 虽然具有较低的推理延迟，但随着空洞率增大，局部采样变得稀疏，导致对细粒度指令模式的刻画能力受限。而 TCN 通过多层空洞卷积堆叠能够建模较长的依赖关系，其分类性能接近本文方法，但其较深的网络结构增加了计算复杂度，不利于实时部署。Inception1D 受限于固定卷积核组合方式，对复杂操作码行为的跨尺度特征交互建模能力不足。MobileNetV2 虽在参数量控制方面表现突出，但其设计初衷偏向空间特征提取，缺乏针对序列依赖的建模设计，因此难以充分捕捉操作码中的长程结构依赖。

与上述方法相比，Res2Net 模块通过通道分组与分层残差连接，在保持较低计算复杂度的同时有效扩大感受野，更契合操作码序列中局部指令模式与长程结构依赖并存的特性。

3.4.3　分类器结构设计实验

针对分类器部分的结构设计，本文进一步分析了不同多层感知机（MLP）结构对模型性能的影响。具体而言，通过对比不同隐藏层数量的 MLP 结构，在保持其余模块不变的前提下评估模型的收敛速度与分类性能。

实验结果如图 7 所示，单隐藏层结构在训练稳定性和分类准确率方面均表现最优。当隐藏层数增加至两层以上时，模型反而出现一定程度的性能下降，表明在当前融合特征维度与样本规模下，过深的全连接结构容易引入冗余参数，削弱模型的泛化能力。综合模型复杂度控制与参数效率的考虑，本文最终采用单隐藏层 MLP 结构（隐藏层 128→输出层），并将 Dropout 比例设为 0.2。该设计在保持模型轻量化的同时，能够实现对融合特征的有效非线性映射。

表 10　　各多尺度卷积模块在操作码特征表征中的综合性能评估

模型	单样本推理时延 / ms	FLOPs/M	分类准确率	F1 值
TCN [37]	1.8	36.25	99.30%	99.29%
DilatedConv[38]	**1.3**	31.47	99.18%	99.18%
Inception1D[39]	1.8	32.18	99.06%	99.02%
MobileNetV2[40]	1.7	**29.28**	99.18%	99.18%
Res2Net(本文)	1.4	30.25	**99.53**%	**99.53**%

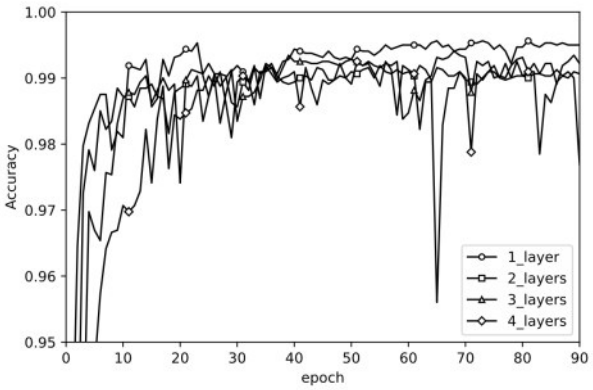


图7　隐藏层数对分类准确率的影响

3.5　对比分析与泛化性实验

3.5.1　综合对比与分析

本文选取的对比方法均为近年来在BIG2015数据集上具有代表性的研究工作，涵盖序列建模方法、图结构建模方法以及基于可视化建模的方法，对比结果如表11所示。

Mosleh等人[46]利用操作码、API调用序列、结构熵和图像纹理等多种静态特征，通过CNN提取类似N-gram的深层特征，在XGBoost上实现了98.90%的准确率。然而，其特征工程高度依赖人工设计，存在特征适应性差、时序依赖关系建模能力不足等局限，制约了其在复杂恶意行为识别中的应用。从建模范式上看，该类方法与本文相似之处在于均利用静态多源特征进行融合建模，但其融合方式更多依赖特征堆叠与人工特征组合，而本文则通过端到端神经网络实现特征自动学习，并在结构设计上强调特征表达压缩与计算复杂度控制。

在深度学习方法探索中，Guo等人[47]提出基于BERT的函数调用图嵌入方法，通过注意力机制联合建模调用图的结构与语义信息，显著提升了对程序行为的理解能力。该方法与本文在技术思路上具有一定相似性，均引入注意力机制刻画跨结构依赖关系。但不同之处在于，其依赖图结构构建及多层聚合操作，模型规模较大，计算路径较长，在面对大规模样本或复杂调用关系时计算开销较高，限制了其在资源受限或实时检测场景中的可行性。Cam等人[48]则提出结合Deep Q-Learning与可解释性分析的恶意软件分类框架。该方法将特征选择与分类过程建模为强化学习问题，通过Q-learning机制动态优化特征子集，同时引入SHAP等方法增强可解释性。其优势在于策略层面的自适应优化能力，但训练过程涉及多阶段策略更新与价值函数迭代，计算路径较长。相比之下，本文未引入强化学习决策层，而是在特征表达阶段通过三元组语义抽象实现结构压缩，并通过轻量化卷积与受控规模注意力机制完成端到端建模，更强调复杂度控制与推理效率。

部分基于可视化的恶意软件分析方法则通过将二进制或字节序列映射为图像，实现端到端建模，降低了对逆向工程的依赖。例如，Chaganti等人[41][41]构建了基于EfficientNetB1的轻量化模型，将参数规模压缩至6.43M，但其训练耗时高达2044秒，难以满足快速迭代和实时部署需求。Wang等人[44]则采用彩色图像编码策略，结合局部区域掩码与非对称的编码器-解码器架构，实现了97.85%的准确率，不适合资源受限环境。Fan等人[42]提出的GCSA-ResNet方法通过协同通道与空间注意力机制增强图像特征表达能力，该方法依赖将字节序列映射为二维图像，并通过深层卷积结构进行特征学习。从方法类型上看，该类图像建模方法与本文同属端到端深度学习框架，但其核心差异在于特征表达路径不同：图像方法通过字节到像素的映射实现结构提

表11　与基于BIG2015数据集的研究对比

研究工作	训练时间	参数量	分类准确率	F1值
Chaganti et al.[41] (2022)	2044s	6.43M	98.58%	98.90%
Fan Y et al.[42] (2025)	-	-	98.50%	-
Zou B et al.[43] (2024)	474s	**114K**	97.20%	92.63%
Wang F et al.[44] (2024)	-	115.1M	97.85%	97.86%
Anand S et al.[45] (2025)	-	180K	97.89%	97.56%
Mosleh M et al[46] (2024)	**74s**	12.93M	98.90%	97.81%
Guo W et al.[47] (2025)	-	-	98.90%	98.87%
Cam N T et al.[48] (2026)	-	-	98.73%	97.19%
本文模型 (2026)	124s	533.5K	**99.53%**	**99.53**%

取，而本文直接在指令序列与API语义层面进行建模，避免字节到图像转换过程中可能造成的指令顺序信息破坏。

综合来看，现有方法在BIG2015数据集上虽已取得较高分类精度，但仍存在一定局限。序列建模方法在上下文依赖刻画和长序列建模效率方面仍有提升空间[49]；基于图像的分析方法虽具备一定结

构表征能力，但字节流到图像的转换会改变原始指令顺序与控制流结构，使函数调用关系和跳转逻辑等关键信息难以完整保留。同时，该类方法的高分辨率输入也增加了计算开销，限制了实际应用场景。相比之下，本文模型在设计上更贴近静态恶意软件分析需求，能够在保证判别能力的同时控制计算开销，更适用于资源受限环境下的部署应用。

3.5.2　企业级场景验证实验

为验证所提方法在真实工业环境中的泛化能力与实用价值，本研究联合领先的安全企业构建了企业级 PE 恶意软件数据集。该数据集覆盖 10 个高活跃恶意软件家族，共包含 13,485 个来自真实攻击事件与长期威胁监测过程的样本。实验采用与基准测试一致的设置，且未进行任何针对特定家族的参数调优，以确保评估结果的客观性和可靠性。

实验结果如表 12 所示，所提出的方法在企业级数据集上取得了 99.41% 的分类准确率。值得注意的是，对于采用混淆技术（如 API 插入或操作码重排）的 Emotet、Sivis 等家族，模型均实现了 100% 的分类准确率，说明融合操作码多尺度特征与静态 API 语义建模的策略，能够在一定程度上缓解 API 插入、操作码重排等常见混淆手段对静态分类模型造成的干扰，证明了其在实际安全防护中的应用价值。但是，模型在 Trickbot 家族上的分类准确率为 73.91%，明显低于其他家族。

图 8 所示的混淆矩阵表明，Trickbot 样本在特征空间中与 Fragtor 家族存在较高重叠。通过对这两类家族提取的操作码序列以及语义链进行分析，本文发现 Trickbot 与 Fragtor 在程序初始化阶段存在一定共性，二者包含模块加载、异常处理及窗口注册相关调用，如 GetModuleHandle、GetProcAddress、LoadLibrary 等。这些共性行为导致两类样本在 API 语义表示的前半部分存在重叠。但在后续执行阶段，Fragtor 更侧重文件操作与进程控制相关调用，而 Trickbot 则包含较多界面交互与资源加载行为，所以两者在功能侧重点上仍然存在

差异。同时，从操作码层面观察发现，Fragtor 样本在初始化阶段呈现较为固定的指令模板，多个样本共享高度相似的 push-sub-mov-call 结构及相对稳定的条件跳转模式，整体结构重复度较高。相比之下，Trickbot 样本内部存在明显变体，部分样本包含混淆型 add、invalid 等操作，其控制流结构表现出更大的离散性。在特征空间中，结构较为一致的 Fragtor 样本更容易形成分布集中的类别表示，而结构差异略大的 Trickbot 样本分布较为分散。

表 12　企业级 PE 数据集各家族分类准确率

家族	分类准确率	样本数
Autoit	100.0%	1633
Barys	98.21%	928
Emotet	100.0%	380
Fragtor	95.74%	312
Fsysna	99.85%	2163
Msil	99.56%	763
Nitol	99.67%	1018
Sivis	100.0%	2372
Stormattack	100.0%	3763
Trickbot	73.91%	153
所有家族	99.41%	13485

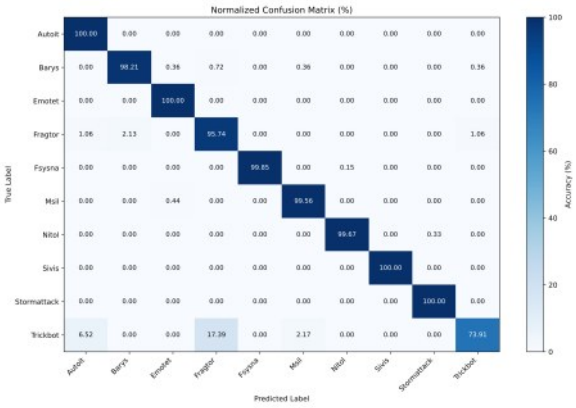


图 8　企业级场景验证实验混淆矩阵图

因此，部分 Trickbot 样本在初始化阶段与 Fragtor 共享相似的指令模板，这些样本的嵌入表示可能会接近 Fragtor 类别，从而被模型判定为 Fragtor。此外，由于 Fragtor 内部方差较小，其样本较少偏离自身类别中心，导致这种单向混淆现象。这种 Trickbot→Fragtor 的单向混淆现象，其根源并非特征缺失，而是初始化阶段的部分共性行为造成的表征塌缩。

基于这一判断，缓解策略应当避免模型过度依赖启动阶段的局部共性，并显式放大两者在后续执行阶段的差异（Fragtor 偏文件/进程控制，Trickbot 偏界面交互/资源加载），因此本文讨论两类改进：

一是在API语义链中引入简单上下文窗口将相邻语义单元组合为短语级Token，从“动作－对象－类别”提升到“局部路径片段”表征，使模型能利用调用顺序与短程依赖来区分“相同启动API但后续路径分叉”的样本；二是引入文件头部/结构统计特征（如节区数量、节区熵、导入表规模等）作为辅助模态，以补足行为语义难以表达的结构差异。但是，上述策略并不能完全消除语义相近家族之间的混淆：局部窗口主要捕捉短程路径信息，对于长程依赖或跨阶段的语义差异无法完全覆盖；结构统计特征也有可能被对抗性填充或其他防护手段所影响，同时它们会增加一些特征提取和计算上的开销。不过它们的价值在于从路径信息和结构先验两个维度缓解特征表征相近的问题，为解决家族间混淆问题提供了可解释的思路。

3.5.3 跨数据集泛化性能评估

为了评估模型在真实开放环境下面对未知家族的泛化能力，本文设计了跨数据集评估实验，分别从BIG2015和企业级数据集中筛选7162个和8184个样本作为训练集与测试集。两个数据集来源于不同的采集环境，其所包含的恶意软件家族完全互斥，且在标签层面存在显著的语义差异，能够较好地模拟真实应用场景中已知家族训练、未知家族测试的开放环境。

由于直接在细粒度家族标签上进行跨数据集评估会受到语义不一致的影响，本文在参考微软恶意软件百科与VirusTotal行为分析报告的基础上，以恶意软件的核心攻击意图作为划分依据，将细粒度家族标签统一映射为Worm（蠕虫）、Botnet（僵尸网络）和Trojan（木马）三类通用行为类别。

在该映射规则下，训练集中的Ramnit与测试集中的Fsysna被归入Worm类，两者均具有明显的自我复制与传播特征；训练集中的Kelihos与Simda，以及测试集中的Stormattack、Nitol、Emotet和Trickbot被统一划分为Botnet类，这些家族虽然在载荷形式和传播方式上存在差异，但其行为核心均表现为依赖命令与控制服务器进行持续通信；此外，训练集中的Gatak、Tracur、Vundo与测试集中的Barys、Fragtor被归为Trojan类，其主要特征为本地隐蔽执行、信息窃取或恶意载荷下载，不具备显式的网络传播行为。

为避免引入语义模糊或描述实现技术的标签，我们剔除了训练集中的Lollipop与Obfuscator.ACY，以及测试集中的Autoit与Msil。这几类标签主要描述实现技术、封装语言或混淆工具属性，而非稳定且可复用的恶意行为语义，难以与三大通用类别形成一致映射。

经过上述标准化处理，最终形成如表13所示的数据集配置：训练集包含7162个样本，覆盖Ramnit等6个已知家族；测试集包含8184个样本，覆盖Stormattack等6个完全未知的家族，在Worm、Botnet与Troja三类高层行为语义上实现了较为严格的对齐，为后续跨数据集泛化评估奠定了基础。从图9所示的实验结果可以看出，尽管训练集与测试集在样本来源、编译环境及家族变种等方面存在差异，但模型在跨数据集三分类评估中仍取得了82.17%的分类准确率，较高的Precision和Recall表明模型在区分Worm、Botnet和Trojan三类行为语义时，预测结果具有较强的确定性，未出现明显的类别塌缩或向单一类别偏置的现象。所以，本文提出的操作码多尺度特征与API三元组语义链融合框架能够有效刻画恶意软件的行为意图，而非仅对特定家族的实现细节进行拟合，具备良好的泛化潜力与实际应用价值。

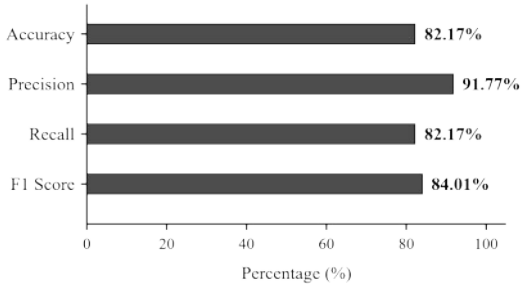


图9 跨数据集泛化实验结果

3.6 可部署性验证实验

为验证所提方法在常见PC处理器上的可部署性，本文在Intel Core i7-13700H CPU（14 cores, 20 threads）环境下对Batch Size=1条件下的部署开销进行评估。实验基于PyTorch CPU推理模式，统计模型加载时间、端到端处理时延以及峰值内存占用。其中，端到端时延根据处理阶段不同分别进行测量。

在模型推理阶段，将已构建的特征张量作为输入，统计神经网络前向传播过程的处理时间。测试结果如表14所示。在单线程CPU环境下，单样本

表 13　跨数据集实验的数据集映射关系及样本数量

通用大类	训练集家族	训练样本数	测试集家族	测试样本数
Worm	Ramnit	1541	Fsysna	2163
Botnet	Kelihos、Simda	3382	Stormattack、Nitol、Emotet、Trickbot	4781
Trojan	Gatak、Tracur、Vundo	2239	Barys、Fragtor	1240
总计	-	7162	-	8184

端到端推理时延中位数为 1.269 ms，90 分位（P90）时延为 1.487 ms，峰值内存占用为 206.20 MB，模型加载时间为 2.553 ms。

同时，为评估全流程特征构建的计算开销，本文针对从原始 ASM 文件输入到特征张量生成的完整过程进行性能测试。该过程包含 opcode 序列和 API 序列提取、三元组映射以及张量化处理。实验数据表明（见表 14），特征提取阶段的端到端时延中位数为 78.982 ms，P90 时延为 310.53 ms，峰值内存占用为 198.55 MB。由于不同 ASM 文件的指令规模存在差异，所以处理耗时呈现出一定的长尾效应。

表 14　端到端环境下的部署开销

阶段	端到端推理时延/ms	峰值内存/MB
模型推理	1.269	206.20
特征提取	78.982	198.55

上述结果表明，在特征提取与模型推理两个阶段中，单样本端到端处理开销可控制在毫秒级，且耗时主要集中于特征提取环节，能够说明本文方法在模型结构层面具备较好的轻量化特性与推理效率。需要强调的是，以上测试均在纯 CPU 环境下完成，未启用 GPU 加速。在常见 PC 处理器配置下，整体处理时延仍处于可接受范围，能够满足实际分类应用对时效性的要求。

此外，本文统计的端到端时延未包含原始可执行文件的反汇编耗时。其原因主要有三点：第一，BIG2015 官方数据集提供 ASM 文件，因此无需执行反汇编；第二，在工程化部署中，反汇编通常作为独立的离线预处理模块运行，其耗时更多受外部反汇编工具与样本规模影响，可通过批量离线生成或并行化处理进行加速；第三，以本文 PE 数据集为例，采用 Python 调用 IDA Pro 接口进行单进程反汇编，吞吐率约为 11 个样本/分钟。基于上述考虑，本文的时延评估主要考虑特征提取和模型推理开销，以客观地刻画本文方法的部署成本与实时性能。

4　结束语

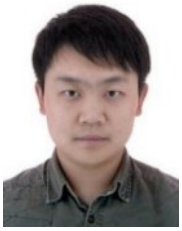
传统静态分类方法在特征表达能力与计算效率之间难以兼顾：单一静态特征难以充分刻画恶意行为，多特征融合方法又往往因模型复杂、开销较大而不利于实际部署。针对这一问题，本文提出了一种面向资源受限场景的轻量化恶意软件分类方法。该方法以反汇编生成的 ASM 文件为统一输入，对操作码序列和静态 API 调用进行联合建模，并从特征表达与网络结构两方面降低冗余、控制复杂度，从而在分类精度与推理时延之间取得较好平衡。

但该方法仍存在一定局限。首先，其对动态行为特征的刻画能力不足，因此在面对反编译保护、代码虚拟化等高级威胁时，分类性能仍有待提升。其次，在数据分布不均衡的情况下，对小样本家族的识别效果仍然有限。

针对上述不足，未来可从以下方向进一步研究：一是探索动态与静态特征的融合机制，结合 API 调用参数、网络流量等动态行为信息，构建混合分类框架，以提升对高级混淆和动态逃逸技术的识别能力；二是针对数据不平衡问题，引入生成对抗网络（GAN）或数据重采样策略，缓解样本不均衡和数据稀缺带来的分类偏差。通过上述方法的协同优化，最终构建具备更强动态适应能力的高性能恶意软件分类框架，为网络安全防护提供更有效的技术支撑。

赵攀（2001– ），男，江苏淮安人，主要研究方向为网络与信息安全、恶意软件分类与检测。

谢雪（1989– ），男，吉林长春人，博士，高级研究员，主要研究方向为多媒体取证、信息隐藏、人工智能安全。



郑文鑫（1987– ），男，山东潍坊人，中级工程师，主要研究方向为计算机图像分析，电子数据取证分析、网络安全溯源分析。

王金伟（1978– ），男，内蒙古呼伦贝尔人，博士，教授，主要研究方向为多媒体版权保护、多媒体取证、多媒体加密和数据认证。

参考文献：

[1] Aboaoja F A, Zainal A, Ghaleb F A, et al. Malware detection issues, challenges, and future directions: A survey[J]. Applied Sciences, 2022, 12(17): 8482.

[2] AV-TEST. Website[EB/OL]. Available: https://portal. av-atlas. org/malware.

[3] Gandotra E, Bansal D, Sofat S. Malware analysis and classification: A survey[J]. Journal of Information Security, 2014, 5(2): 56-64.

[4] Elhadi A A E, Maarof M A, Barry B I A, et al. Enhancing the detection of metamorphic malware using call graphs[J]. Computers & Security, 2014, 46: 62-78.

[5] Shabtai A, Moskovitch R, Feher C, et al. Detecting unknown malicious code by applying classification techniques on opcode patterns[J]. Security Informatics, 2012, 1(1): 1-22.

[6] Shaid S Z M, Maarof M A. Malware behaviour visualization[J]. Jurnal Teknologi, 2014, 70(5): 25-33.

[7] Kakisim A G, Gulmez S, Sogukpinar I. Sequential opcode embedding-based malware detection method[J]. Computers & Electrical Engineering, 2022, 98: 107703.

[8] Khalilian A, Nourazar A, Vahidi-Asl M, et al. G3MD: Mining frequent opcode sub-graphs for metamorphic malware detection of existing families[J]. Expert Systems with Applications, 2018, 112: 15-33.

[9] Li C, Cheng Z, Zhu H, et al. DMalNet: Dynamic malware analysis based on API feature engineering and graph learning[J]. Computers & Security, 2022, 122: 102872.

[10] Gibert D, Mateu C, Planes J. HYDRA: A multimodal deep learning framework for malware classification[J]. Computers & Security, 2020, 95: 101873..

[11] Chen T, Zeng H, Lv M, et al. CTIMD: Cyber threat intelligence enhanced malware detection using API call sequences with parameters [J]. Computers & Security, 2024, 136: 103518.

[12] Li C, Lv Q, Li N, et al. A novel deep framework for dynamic malware detection based on API sequence intrinsic features[J]. Computers & Security, 2022, 116: 102686.

[13] Yuan C, Cai J, Tian D, et al. Towards time evolved malware identifica-

tion using two-head neural network[J]. Journal of Information Security and Applications, 2022, 65: 103098.

[14] Jeon S, Moon J. Malware-detection method with a convolutional recurrent neural network using opcode sequences[J]. Information Sciences, 2020, 535: 1-15.

[15] Sewak M, Sahay S K, Rathore H. LSTM hyper-parameter selection for malware detection: Interaction effects and hierarchical selection approach[C]//2021 International Joint Conference on Neural Networks (IJCNN). IEEE, 2021: 1-9.

[16] Hao J, Luo S, Pan L. EII-MBS: Malware family classification via enhanced adversarial instruction behavior semantic learning[J]. Computers & Security, 2022, 122: 102905.

[17] Darem A, Abawajy J, Makkar A, et al. Visualization and deep-learning-based malware variant detection using opcode-level features[J]. Future Generation Computer Systems, 2021, 125: 314-323.

[18] Shaukat K, Luo S, Varadharajan V. A novel method for improving the robustness of deep learning-based malware detectors against adversarial attacks[J]. Engineering Applications of Artificial Intelligence, 2022, 116: 105461.

[19] Demirkıran F, Çayır A, Ünal U, et al. An ensemble of pre-trained transformer models for imbalanced multiclass malware classification[J]. Computers & Security, 2022, 121: 102846.

[20] Vinayakumar R, Alazab M, Soman K P, et al. Robust intelligent malware detection using deep learning[J]. IEEE Access, 2019, 7: 46717-46738.

[21] Huo D, Li X, Li L, et al. The application of 1D-CNN in Microsoft malware detection[C]//2022 7th International Conference on Big Data Analytics (ICBDA). IEEE, 2022: 181-187.

[22] Do Xuan C, Huong D T. A new approach for APT malware detection based on deep graph network for endpoint systems[J]. Applied Intelligence, 2022, 52(12): 14005-14024.

[23] Bensaoud A, Kalita J. CNN-LSTM and transfer learning models for malware classification based on opcodes and API calls[J]. Knowledge-Based Systems, 2024, 290: 111543.

[24] Xuan B, Li J, Song Y. BiTCN-TAEfficientNet malware classification approach based on sequence and RGB fusion[J]. Computers & Security, 2024, 139: 103734.

[25] Babu S, Singh V. Bd-mdlc: Behavior description-based enhanced malware detection for Windows environment using Longformer classifier [J]. Computers & Security, 2024, 146: 104031.

[26] Xu Z, Fang X, Yang G. MalBERT: A novel pre-training method for malware detection[J]. Computers & Security, 2021, 111: 102458.

[27] Zhou Z, Li Y, Li J, et al. GAN-Siamese Network for cross-domain vehicle re-identification in intelligent transport systems[J]. IEEE Transactions on Network Science and Engineering, 2023, 10(5): 2779-2790.

[28] Hsiao S C, Kao D Y, Liu Z Y, et al. Malware image classification using one-shot learning with Siamese networks[C]//Proceedings of the 23rd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems. Procedia Computer Science, 2019, 159: 1863-1871.

[29] Gao S H, Cheng M M, Zhao K, et al. Res2Net: A new multi-scale backbone architecture[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019, 43(2): 652-662.

[30] Sikorski M, Honig A. Practical malware analysis: The hands-on guide to dissecting malicious software[M]. No Starch Press, 2012.

[31] Rudd E M, Rozsa A, Günther M, et al. A survey of stealth malware attacks, mitigation measures, and steps toward autonomous open world solutions[J]. IEEE Communications Surveys & Tutorials, 2017, 19(2): 1145-1172. DOI: 10.1109/COMST.2016.2636078.

[32] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in Neural Information Processing Systems, 2017, 30.

[33] Kebede T M, Djaneye-Boundjou O, Narayanan B N, et al. Classification of malware programs using autoencoders based deep learning architecture and its application to the Microsoft malware classification challenge (BIG 2015) dataset[C]//2017 IEEE National Aerospace and Electronics Conference (NAECON). IEEE, 2017: 70-75.

[34] Cho K, Van Merriënboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation[J]. arXiv preprint arXiv:1406.1078, 2014.

[35] Graves A, Schmidhuber J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures[J]. Neural Networks, 2005, 18(5-6): 602-610.

[36] He P, Liu X, Gao J, et al. DeBERTa: Decoding-enhanced BERT with disentangled attention[J]. arXiv preprint arXiv:2006.03654, 2020.

[37] Bai S, Kolter J Z, Koltun V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling[J]. arXiv preprint arXiv:1803.01271, 2018.

[38] Yu F, Koltun V. Multi-scale context aggregation by dilated convolutions[J]. arXiv preprint arXiv:1511.07122, 2015.

[39] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015: 1-9.

[40] Sandler M, Howard A, Zhu M, et al. MobileNetV2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.

[41] Chaganti R., Ravi V., Pham T. D., 2022. Image-based malware representation approach with EfficientNet convolutional neural networks for effective malware classification.

[42] Fan Y, Zhang K, Zheng B, et al. GCSA-ResNet: a deep neural network architecture for Malware detection[J]. Scientific Reports, 2025, 15(1): 24098.

[43] Zou B, Cao C, Wang L, et al. FACILE: A capsule network with fewer capsules and richer hierarchical information for malware image classification[J]. Computers & Security, 2024, 137: 103606.

[44] Wang F, Shi X, Yang F, et al. Malsort: Lightweight and efficient image-based malware classification using masked self-supervised framework with swin transformer[J]. Journal of Information Security and Applications, 2024, 83: 103784.

[45] Anand S, Mitra B, Dey S, et al. Malite: Lightweight malware detection and classification for constrained devices[J]. IEEE transactions on emerging topics in computing, 2025.

[46] Mosleh M R B, Sharifian S. An efficient cloud-integrated distributed deep neural network framework for IoT malware classification[J]. Future Generation Computer Systems, 2024, 157: 603-617.

[47] Guo W, Du W, Yang X, et al. MalHAPGNN: An Enhanced Call Graph-Based Malware Detection Framework Using Hierarchical Attention Pooling Graph Neural Network[J]. Sensors, 2025, 25(2): 374.

[48] Cam N T, Huy T M, Tin N T. Malware classification using deep neural networks with Deep Q-Learning and eXplainable artificial intelligence [J]. Engineering Applications of Artificial Intelligence, 2026, 166:

113622.

[49] 王金伟,陈正嘉,谢雪等.基于Ngram-TFIDF的深度恶意代码可视化分类方法[J].通信学报,2024,45(06):160-175. DOI: 10.11959/j.issn.1000-436x.2024115.

[作者简介]

陈治国（1988–），男，江苏南京人，博士，副教授，主要研究方向为信息安全、分布式算法、云计算和虚拟现实。