软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007253] [CSTR: 32375.14.jos.007253]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于 BERT 与自编码器的概念漂移恶意软件分类优化*

赵浩钧[1,2,3,5], 邹德清[1,2,3,5], 薛文杰[1,2,3,5], 吴月明[6], 金 海[1,2,4,7]

[1](大数据技术与系统国家地方联合工程研究中心, 湖北 武汉 430074)

[2](服务计算技术与系统教育部重点实验室, 湖北 武汉 430074)

[3](大数据安全湖北省工程研究中心, 湖北 武汉 430074)

[4](集群与网格计算湖北省重点实验室, 湖北 武汉 430074)

[5](华中科技大学 网络空间安全学院, 湖北 武汉 430074)

[6](南洋理工大学 计算机与数据科学学院, 新加坡 639798)

[7](华中科技大学 计算机科学与技术学院, 湖北 武汉 430074)

通信作者: 邹德清, E-mail: deqingzou@hust.edu.cn

摘　要: 软件概念漂移指同类型软件的软件结构和组成成分会随着时间的推移而改变. 在恶意软件分类领域, 发生概念漂移意味着同一家族的恶意样本的结构和组成特征会随时间发生变化, 这会导致固定模式的恶意软件分类算法的性能会随时间推移而发生下降. 现有的恶意软件静态分类研究方法在面临概念漂移场景时都会有显著的性能下降, 因此难以满足实际应用的需求. 针对这一问题, 鉴于自然语言理解领域与二进制程序字节流分析领域的共性, 基于 BERT 和自定义的自编码器架构提出一种高精度、鲁棒的恶意软件分类方法. 该方法首先通过反汇编分析提取执行导向的恶意软件操作码序列, 减少冗余信息; 然后使用 BERT 理解序列的上下文语义并进行向量嵌入, 有效地理解恶意软件的深层程序语义; 再通过几何中位数子空间投影和瓶颈自编码器进行任务相关的有效特征筛选; 最后通过全连接层构成的分类器输出分类结果. 在普通场景和概念漂移场景中, 通过与最先进的 9 种恶意软件分类方法进行对比实验验证所提方法的实际有效性. 实验结果显示: 所提方法在普通场景下的分类 $F1$ 值达到 99.49%, 高于所有对比方法, 且在概念漂移场景中的分类 $F1$ 值比所有对比方法提高 10.78%–43.71%.

关键词: 恶意软件静态分析; 概念漂移; 鲁棒性优化

中图法分类号: TP311

中文引用格式: 赵浩钧, 邹德清, 薛文杰, 吴月明, 金海. 基于BERT与自编码器的概念漂移恶意软件分类优化. 软件学报. http://www.jos.org.cn/1000-9825/7253.htm

英文引用格式: Zhao HJ, Zou DQ, Xue WJ, Wu YM, Jin H. Optimization of Concept Drift Malware Classification Based on BERT and Autoencoder. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7253.htm

Optimization of Concept Drift Malware Classification Based on BERT and Autoencoder

ZHAO Hao-Jun[1,2,3,5], ZOU De-Qing[1,2,3,5], XUE Wen-Jie[1,2,3,5], WU Yue-Ming[6], JIN Hai[1,2,4,7]

[1](National Engineering Research Center for Big Data Technology and System, Wuhan 430074, China)

[2](Key Laboratory of Services Computing Technology and System, Ministry of Education, Wuhan 430074, China)

[3](Hubei Engineering Research Center on Big Data Security, Wuhan 430074, China)

[4](Hubei Key Laboratory of Cluster and Grid Computing, Wuhan 430074, China)

[5](School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China)

[6](School of Computing and Data Science, Nanyang Technological University, Singapore 639798, Singapore)

[7](School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

* 基金项目: 国家自然科学基金面上项目 (62172168)
收稿时间: 2023-12-09; 修改时间: 2024-04-28; 采用时间: 2024-07-13; jos 在线出版时间: 2024-12-04

Abstract: Software concept drift means that the structure and composition of the same type of software will change over time. In malware classification, concept drift means that the structure and composition characteristics of malware samples from the same family can change over time. This will cause a decline in the performance of fixed-mode malware classification algorithms over time. Existing methods for static malware classification experience significant performance degradation when faced with concept drift scenarios, making it difficult to meet the needs of practical applications. To address this problem, given the commonalities between natural language understanding and binary byte stream analysis, a highly accurate and robust malware classification method is proposed based on BERT and a custom autoencoder architecture. This method extracts execution-oriented malware opcode sequences through disassembly analysis to reduce redundant information. Then, it uses BERT to understand the contextual semantics of the sequences and perform vector embedding to effectively understand the deep program semantics of the malware samples. It also screens effective task-related features through the geometric median subspace projection and bottleneck autoencoders. Finally, a classifier composed of fully connected layers is used to output the classification results. The practical effectiveness of the proposed method is validated through comparative experiments with nine state-of-the-art malware classification methods in both normal and concept drift scenarios. Experimental results show that the proposed method achieves an $F1$ score of 99.49% in normal scenarios, outperforming those nine methods. Moreover, in concept drift scenarios, the $F1$ score is improved by 10.78% to 43.71% compared to the nine methods.

Key words: malware static analysis; concept drift; robust optimization

作为全球使用最广泛的操作系统, Windows 由于其开放的开发环境和复杂的软件来源, 特别容易受到恶意软件的攻击. Windows 恶意软件分类有助于提供准确的威胁识别和分类, 帮助加强计算机系统的安全防御, 促进恶意软件研究和分析以及推动威胁情报的共享与合作, 是保护计算机系统和用户安全的重要研究内容. 在静态分析方法中, 相较于传统基于规则的方法, 基于学习的恶意软件分类可以从大量样本中学习恶意软件家族的潜在特征, 具有更强的自适应性和泛化能力, 且不需要专家知识制定复杂的匹配规则就能达到较高的准确率. 近些年, 基于学习的 Windows 恶意软件静态分类逐渐成为热点研究方向.

基于学习的 Windows 恶意软件静态分类方法通过静态分析恶意软件的二进制文件, 提取文件特征 (例如特征字符串、API 调用等), 然后使用机器学习算法进行模型训练和预测. 后来随着深度学习的发展, 一些研究工作[1–3]通过将恶意软件二进制文件转换成图片或文本的形式, 结合图像处理或自然语言处理领域 (NLP) 的深度学习模型进行家族分类预测. 然而, 现有的基于学习的恶意软件分类方法往往缺乏对程序语义信息的深度挖掘, 在复杂环境中表现的准确性和稳定性存在不足, 尤其是在概念漂移场景中, 容易出现分类性能大幅下降的问题. 通过对最先进的 9 种 Windows 恶意软件分类工作进行大规模的评估实验, Ma 等人[4]发现现有方法在概念漂移场景中的性能下降巨大, 难以满足工业场景的实际应用需求. 普通场景下表现优秀的分类模型在概念漂移场景下性能下降幅度甚至可以达到 69.62%. 针对上述问题, 本文提出了一种基于学习的新型恶意软件分类方法, 通过结合反汇编分析和自然语言处理中的字节流分析, 实现对恶意软件的深层语义进行挖掘, 并对提取特征进行去偏和降维, 提取与任务相关的有效特征, 提高模型表现的稳定性, 以在概念漂移场景中达到更好的表现.

具体来说, 本文主要面临两个主要挑战.

(1) 如何处理恶意二进制文件并将其转换为适合学习模型的输入.

(2) 如何设计有效的分析模式来降低概念漂移对模型分类性能的负面影响.

针对第 1 个挑战, 需要考虑将恶意软件转化为特征有效且适合学习算法处理的形式, 并且能够深刻体现恶意软件的语义特征. 一些工作[5–7]通过分析程序的函数调用图或控制流图以进行恶意软件分类, 其分类的准确性很大程度上取决于特征提取和利用的方式. 然而手动定义的特征提取会遗失许多程序潜在的上下文语义关系. 因此应该考虑让学习算法更多地处理原始特征信息, 利用深度学习算法来理解程序的上下文语义, 让分类器更加类似于端到端的结构. 一些基于二进制流分析的工作[8,9]将二进制文件转换为字节序列或灰度图像, 然后使用自然语言处理模型或图像分类模型对其进行分析. 尽管这些工作利用学习模型来理解程序的语义信息, 但它们的性能并不突出. 这是由于恶意程序二进制流中包含大量与程序行为无关的冗余信息 (例如通用文件结构和数据信息), 此外文本和图像的构造也可能为特征数据加入本不应该存在的相关性. 针对上述问题, 本文使用反汇编技术和字节流分析技术相结合的方法, 以程序入口为起点获得执行导向的操作码序列, 以最大限度保留程序行为信息, 然后使用

BERT 模型对操作码序列进行上下文关系挖掘和特征向量嵌入, 以获得包含深层上下文语义关系信息的特征表示形式.

针对第 2 个挑战, 需要考虑为恶意软件分类的特征处理和模型训练过程进行优化. 虚假相关性是自然语言理解 (NLU) 领域的一个重要问题[10,11], 即学习模型学习到的特征之间存在虚假的相关性, 错误地将无关的特征与目标任务相关联. 具体表现为模型在分布内数据集上表现为高性能, 而在分布外数据集上性能较差. 通过去偏方法能够减少虚假相关性, 迫使学习模型学习与任务更相关的特征. 概念漂移前后的数据可以看作两个数据集, 因此概念漂移场景下的恶意软件分类与 NLU 领域中的消除虚假相关性问题存在相似性. 鉴于文件二进制流分析与自然语言文本分析、概念漂移性能下降问题与 NLU 虚假相关性问题的相似性, 考虑将 NLU 领域的去偏方法应用于恶意软件分类的特征处理和模型训练过程, 通过消除特征与目标任务的虚假依赖, 减少模型在概念漂移前后的性能下降.

针对上述问题, 本文提出一种基于 BERT 和自定义自编码器架构的恶意软件分类方法, 可以在普通场景和概念漂移场景下都实现较好的性能. 首先, 使用反汇编工具对恶意软件二进制文件进行反汇编分析, 以程序入口点为起点分析程序可能的执行路径, 并根据执行路径提取指令操作码, 以操作码序列的形式存储. 然后将反汇编分析得到的操作码序列, 输入到基于 BERT 的滑动窗口中进行词嵌入处理, 得到多段特征向量并进行拼接, 生成代表该软件的特征向量. 通过瓶颈自编码器结合几何中位数子空间投影的特征降维方法, 提取出与分类目标任务最相关的特征, 然后进行分类. 本文实现了基于 BERT 和自编码器的恶意软件分类原型系统 MCBA (malware classifier with BERT and autoencoder), 并在与研究[3]相同的实验条件下, 与 9 种最先进的恶意软件分类工作进行对比, 评估方法在普通场景和概念漂移场景下的准确性和鲁棒性. 实验证明 MCBA 的准确性和稳定性高于其他 9 种方法, 特别是在概念漂移场景中, MCBA 的性能表现高于其他方法 10.78%–43.71%.

本文的主要贡献如下.

(1) 提出一种基于程序语义和字节流分析的恶意软件分类方法, 可以更加准确的捕获程序的上下文语义.

(2) 引入 NLP 的去偏算法和优化方法, 通过增加自定义的自编码器和几何中位数子空间投影, 对恶意软件的特征提取进行优化, 增强恶意软件分类模型的泛化能力, 使分类方法在概念漂移场景下具有更稳定的性能.

(3) 设计并实现高准确性和鲁棒性的恶意软件分类原型系统 MCBA, 并与 9 种最先进的相关工作进行普通场景和概念漂移场景下的对比实验. 实验结果证明 MCBA 有更高的分类准确性和更强的鲁棒性.

本文第 1 节介绍 Windows 恶意软件分类的相关方法和研究现状. 第 2 节介绍本文所需的背景知识和威胁模型. 第 3 节介绍本文构建的基于 BERT 和自编码器的恶意软件分类原型系统. 第 4 节通过与 9 种最先进相关工作进行对比实验, 在准确性、稳定性和鲁棒性上验证所提方法的有效性. 最后总结全文, 并展望未来工作.

1 相关工作

现有的基于学习的静态恶意软件分类方法主要可以分为 3 类: 基于图像转换的方法、基于字节流分析的方法、基于反汇编分析的方法.

1.1 基于图像转换的恶意软件分类

基于图像转换的方法通过将恶意软件二进制文件转换为灰度图像, 然后使用图像处理领域的分类模型对灰度图进行分类, 从而达到对恶意软件进行家族分类的目的. Nataraj 等人[12]将恶意软件分类与图像分析技术结合起来, 将二进制文件转换为灰度图像, 其中每个字节的值对应灰度图中的像素灰度值, 然后使用 K 邻近算法 (KNN) 对图像进行分类. 后来, Kancherla 等人[13]优化了图像处理方法, 采用 SVM 算法结合 Gabor 滤波器进行特征提取和分类工作. 然而由于提取复杂纹理特征的开销较高, 方法不够高效难以实际应用. 随着深度学习模型在图像分类领域的发展, 基于图像转换的恶意软件分类方法的性能表现也越来越好. 例如, 卷积神经网络 (CNN) 已广泛应用于各种恶意软件分类工作[14–19], 并展现出优秀的性能. 基于图像转换的方法直接处理原始二进制文件, 不需要特定的专业知识来提取特征, 并且有许多经过广泛验证的高性能图像分类算法可以迁移应用.

基于图像转换的方法存在很多缺点, 例如将二进制恶意样本转换为图像不可避免需要引入新的超参数 (例如图像宽度), 这可能会在像素之间引入原本样本特征中不存在的空间相关性, 从而影响分类效果[1]. 此外, 由于恶意软件被整体转换为图像, 其中必然会包含许多冗余信息, 这会使分类器难以识别出与任务目标最相关的特征.

1.2 基于字节流分析技术的恶意软件分类

基于字节流分析的方法将恶意软件的二进制流视为序列信息, 并使用序列模型 (例如 NLP 中的学习模型) 对其进行处理, 然后进行分类. Moskovitch 等人[20]提出基于文本分类技术的恶意软件分类方法. 通过对训练数据提取 *n*-gram, 然后根据自定义的文档频率分数选择最佳的 5 500 个词作为特征, 然后使用学习模型进行分类. Jain 等人[21]使用类别文档频率技术来减少特征空间大小并改进分析方法. Zhang 等人[22]利用信息增益来过滤 *K* 个 *n*-gram 作为特征, 但随着 *n* 值的增加其计算成本呈指数增长. Raff 等人[9]将整个二进制文件作为输入, 使用端到端的浅层卷积神经网络模型进行恶意软件分类. 但由于其对内存的需求较大, 因此处理能力受到限制. Qiao 等人[8]将恶意样本的二进制流按字节取值, 转换为 256 个单词并结合 Word2Vec 模型[23]将程序嵌入为词嵌入矩阵, 然后使用多层感知器 (MLP) 进行恶意软件分类.

基于字节流序列的分析方法在一定程度上考虑了程序的上下文信息, 但由于程序并不总是顺序执行而是不断跳转的, 因此二进制流的上下文信息并不总是相关的. 此外基于字节流分析的方法还需要消耗大量系统资源, 因为恶意软件字节序列可能是非常庞大且冗余的[23], 这阻碍了恶意软件关键特征提取和复杂环境的分析稳定性.

1.3 基于反汇编特征的恶意软件分类

基于反汇编的方法通过对恶意软件二进制文件进行反汇编分析, 从静态反汇编结果中人工提取特征信息然后用于恶意软件家族分类. Ahmadi 等人[24]开发了一个多模态恶意软件分类系统, 使用从 50 万个恶意软件样本中提取的 API 调用频率进行分类. SimHash[25]和 MinHash[26]哈希算法也被尝试用于将操作码序列投影转换为向量, 然后将其转化为图像进行分类. Awad 等人[27]将每个反汇编文件的操作码序列视为一个文档, 并使用 Word2Vec 生成文档的向量表示, 然后使用词移距离 (WMD)[28]和 K 近邻算法对表征恶意样本的文档进行分类. 然而, 这些方法缺乏对程序行为信息的关注, 缺少对程序语义的挖掘. Kinable 等人[6]使用图匹配技术计算两个程序的函数调用图 (FCG) 之间的相似度得分, 并将其作为恶意软件分类的依据. 后来, Kong 等人[7]将恶意软件抽象为属性函数调用图 (AFCG), 以更细粒度的方式进行相似距离匹配. 然而, 这些方法都是计算密集型的, 而且泛化能力不够强. Hassen 等人[5]采用 MinHash 算法对相似恶意软件的函数调用图进行聚类, 然后将图特征转换为向量并使用深度学习算法进行分类. 类似地, Yan 等人[29]对从反汇编文件中提取的 AFCG 进行特征转换然后利用深度图卷积神经网络 (DGCNN)[30]进行分类, 并表现出良好的性能.

然而, 基于反汇编分析的方法需要专业领域知识和人工指导, 如汇编语言和反汇编分析方法. 此外, 基于反汇编的方法分类性能很大程度上受到从反汇编结果中人工选择的特征向量的影响, 因此很难发现潜在的程序上下文语义信息.

2 背景知识和威胁模型

本文的主要目标是通过对恶意软件二进制字节流进行反汇编分析, 结合自然语言处理技术进行深层程序上下文语义理解和任务相关的有效特征提取来实现高准确率和鲁棒的恶意软件分类, 优化分类方法在概念漂移场景下的表现. 所提方法用到的模型和算法理论主要基于 BERT、瓶颈自编码器和几何中位数子空间. 下面就相关知识和威胁模型予以介绍.

2.1 BERT (Transformer 的双向编码器表示)

BERT 是一种用于自然语言处理 (NLP) 的深度学习模型. 它是一种基于转换器 (Transformer) 架构的预训练文本处理模型. Transformer 架构是一种神经网络, 能够并行处理输入序列, 在处理长文本序列的 NLP 任务中表现出优秀的性能. 相较于传统的语言模型从左至右顺序处理自然语言, BERT 采用了双向编码的方法, 能够同时利用左

右两侧的上下文信息, 这使得模型能够考虑句子中蕴含的上下文信息, 更好地理解文本语义和关系. BERT 通过两个无监督学习任务在大型文本语料库上进行预训练: 遮盖 (masked language model, MLM) 和下一句预测 (next sentence prediction, NSP). 在 MLM 任务中, 部分输入被随机屏蔽, 模型经过训练以根据周围上下文预测屏蔽标记. 在 NSP 任务中, 模型通过训练来预测给定文档中的两个句子是否连续. 此外, BERT 也可以直接用于对单个句子进行向量嵌入, 并输出代表句子整体特征的 [CLS] 向量. 预训练后, BERT 模型可以通过微调来适应各种下游任务, 例如情感分析、文本分类、句子关系判断和命名实体识别等. 得益于高质量的上下文词嵌入和对文本上下文含义的理解, BERT 在许多 NLP 基准测试上都取得了最佳性能的表现, 并广泛应用于工业界和学术界的各种 NLP 任务.

在恶意软件分析中, 鉴于二进制文件的字节流序列与自然语言文本序列之间的相似性, 许多工作将二进制流转化为文本进行分析 (例如, 将每个恶意软件二进制文件视为 256 个单词 (0x00–0xFF) 的语料库) 并使用 NLP 模型进行处理.

2.2 瓶颈自编码器 (bottleneck autoencoder)

瓶颈自编码器是一种用于特征提取或数据压缩的无监督神经网络模型. 它是自编码器的一种变体, 其核心思想是通过将输入数据压缩到一个较低维度的编码空间, 然后再将其解码回原始数据空间, 从而学习到数据的紧凑表示, 捕获输入数据的关键特征.

瓶颈自编码器包含一个编码器和一个解码器, 通过多层神经网络构建. 编码器负责将输入数据映射到低维的瓶颈层, 该层称为瓶颈特征或隐藏特征. 通过限制瓶颈层的维度, 瓶颈自编码器迫使模型提取输入数据中最重要和最相关的特征, 并且通过解码器的重构过程尽量还原原始数据. 瓶颈自编码器的训练过程即最小化原始数据与重构数据的重构误差的过程, 这样的压缩和解压缩过程有助于模型去除输入数据中的噪声和冗余信息, 提高数据表示的紧凑性和鲁棒性.

瓶颈自编码器在许多领域中具有广泛应用, 包括特征学习与提取、数据降维、异常检测等. 通过学习到紧凑的数据表示, 瓶颈自编码器能够减少数据的维度, 去除冗余信息, 提取主要特征, 并在一定程度上提高模型的鲁棒性和泛化能力, 从而为后续的分类、聚类和生成等任务提供更好的输入.

2.3 几何中位数子空间 (geometric median subspace)

几何中位数子空间是一种用于估计数据样本所在子空间的方法. 它是多元数据分析中的一种技术, 旨在找到一个子空间, 使得样本到该子空间上的投影之和最小. 在高维数据处理中, 通过寻找高维数据对应的低维子空间, 可以获得数据的变化方向和潜在相关性结构, 从而实现降维、特征提取或聚类等任务目标.

具体而言, 几何中位数子空间的目标是寻找一个 k 维子空间, 使得所有样本在该子空间的投影到其他样本投影的欧氏距离之和最小. 这表明样本在该子空间上的分散程度最小, 因此可以表达最密集的核心信息.

几何中位数子空间在降维、异常检测、子空间聚类等领域具有广泛的应用. 它可以用于发现数据的主要变化模式、挖掘关联性结构以及提取具有代表性的子空间特征. 这种方法对于高维数据的分析具有重要意义, 可以帮助揭示数据的内在结构和重要特征.

2.4 概念漂移 (concept drift)

在预测分析和机器学习中, 概念漂移是指目标变量的统计特性随着时间的推移以不可预见的方式变化的现象[31]. 在恶意软件分类任务中概念漂移的表现是, 同家族的恶意样本会随着时间的推移而发生程序组成和结构上的变化, 使得同家族新出现的样本与之前的样本特征发生改变, 从而使先前训练的分类模型无法准确地将新出现的恶意软件样本进行分类, 影响分类模型的性能. 概念漂移是机器学习和 Windows 恶意软件分类任务中一个现实且关键的问题. 在实际应用中, 恶意软件分类任务经常会处于概念漂移场景中, 现有的分类模型的性能会受到概念漂移的影响而不能满足应用的需求.

2.5 威胁模型

在工业场景中, 恶意软件分类器的性能经常会受到概念漂移的影响. 由于恶意软件编程语言和攻击者对抗性

攻击方法的变化, 同一家族的恶意软件的组成结构和行为特征会随着时间的推移而演变. 在概念漂移场景中, 在旧恶意样本数据上训练的分类模型在分析新恶意样本时表现出性能下降, 这为实际应用带来了严重阻碍. 本文的目标是通过结合反汇编分析和自然语言处理方法, 利用深度学习算法深入理解恶意软件的内在语义特征, 以此增强分类模型的鲁棒性, 减少概念漂移为分类模型带来的性能下降.

为了更加清晰地描述方法和实验, 在本文中不考虑软件加壳对静态分析带来的影响, 并对实验数据进行了脱壳和查壳处理. 软件加壳会改变恶意软件字节流分析的结果, 但由于自动脱壳技术已经相对成熟, 所以本文不作单独讨论.

3 基于 BERT 和自编码器的恶意软件分类

本文结合反汇编技术、自然语言处理技术和特征降维提取方法, 提出一种基于 BERT 和自编码器的恶意软件分类方法并实现原型系统 MCBA. 该系统由 3 部分组成, 系统框架图如图 1 所示.

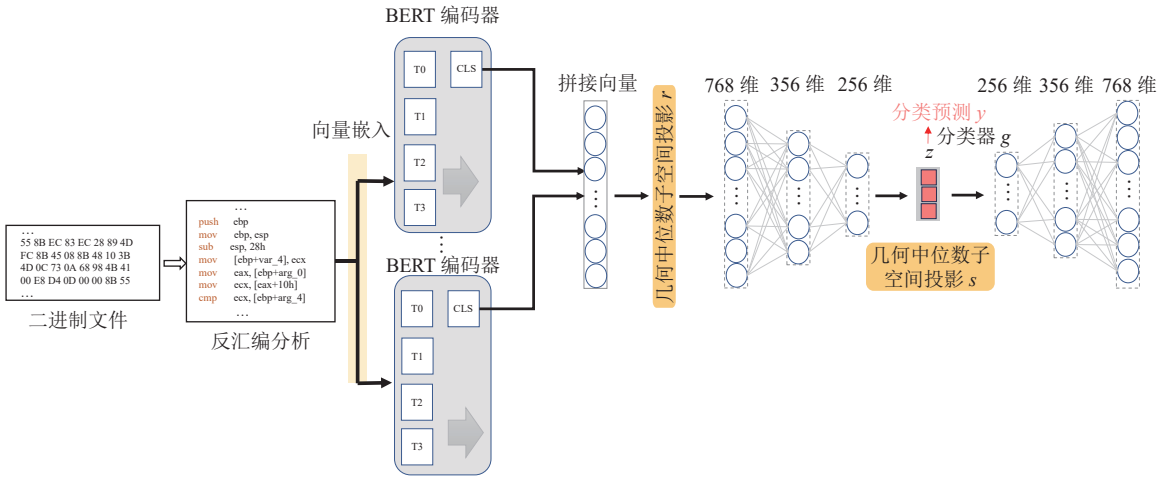


图 1 基于 BERT 和瓶颈自编码器的恶意软件分类模型系统架构

首先, 对目标二进制恶意程序进行快速扫描和反汇编分析. 与其他基于字节流分析的方法不同, MCBA 先通过反汇编分析提取出代表程序语义和行为的反汇编操作码, 过滤掉大部分冗余的程序通用架构和数据信息.

然后, 系统将操作码视为自然语言单词, 将代表该程序特征的操作码序列组合成一个近似自然语言的句子. BERT 接收到代表程序信息的句子并将其作为一个整体进行分词和词向量嵌入, 输出代表整个句子特征的 [CLS] 向量.

接下来, 通过几何中位数子空间投影和瓶颈自编码器对代表程序整体特征的 [CLS] 向量进行降维和关键信息提取, 以得到与分类任务最相关的特征表示形式. 通过训练, 确定几何中位数子空间的投影矩阵以及瓶颈自编码器的网络参数. 最终通过分类器完成恶意软件分类任务.

3.1 基于反汇编的执行导向的操作码提取

为了去除恶意程序的二进制流中的冗余信息, 提取包含语义等关键信息的原始数据供后续分类模型使用, 本文基于 IDA Pro 为 MCBA 设计了面向程序语义的静态分析模块. 现有基于字节流的分类方法通常直接将整个二进制文件的二进制流以 8 位为单位转换为取值为 0–255 的字节流. 这种字节流转换在一定程度上可以反映恶意程序的整体特征和部分程序上下文关系. 然而, 程序在执行过程中, 指令并不总是按地址顺序执行的. 当执行跳转指令时, 传统方法将失去对上下文语义的捕获. 此外, 标准的二进制文件包含许多与恶意软件特征无关的信息. 冗余的字节流信息无疑会为后续分析带来开销和准确性上的问题.

与一般方法不同, MCBA 不直接对原始二进制流进行数值转换, 而是使用基于反汇编的执行导向的操作码提

取方法. MCBA 对原始二进制程序进行反汇编分析, 舍弃一般的文件结构信息、各种数据存储信息、指令操作数信息等, 只保留对指令操作码的关注. 这是因为操作码体现了当前指令的行为, 能够从细粒度反映出程序的行为特点, 且操作码本身也更接近自然语言.

与动态分析实际执行程序不同, 静态分析通常很难分析一个程序真实执行路径. 尽管如此, 通过分析静态信息依然可以提取出可能的控制流方向, 在一定程度上可以反映程序的行为特征. 为了尽量地反映程序的执行逻辑, MCBA 通过附加面向行为的分析模块, 按照程序可能的执行路径提取操作码序列, 以捕获比按地址顺序分析的方法更多的上下文语义, 如图 2 所示.

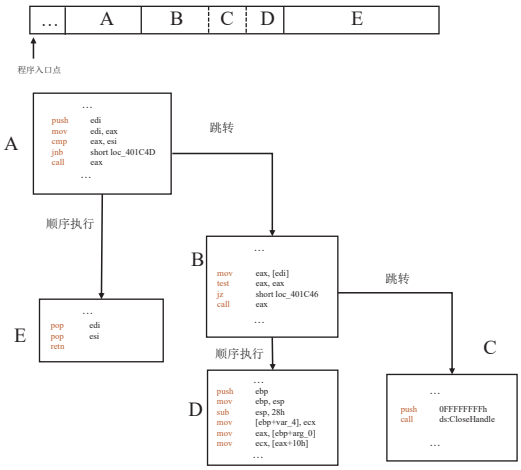


图 2　执行导向的操作码序列提取

MCBA 首先扫描并存储目标程序的入口点. 对于可能有多个入口点的程序, MCBA 一次性存储所有入口点, 然后对所有入口点依次分析. MCBA 从入口点地址开始, 按执行顺序分析接下来的所有指令信息, 并存储操作码助记符. 当当前指令为跳转指令时, MCBA 获取指令的操作数对象 (即跳转地址) 并进行重定位后继续分析指令序列, 以尽可能按照指令的执行顺序进行分析. 特别地, 在每一次重定位前, MCBA 将当前指令的地址存储在一个动态维护的返回地址队列中, 以方便在分析完重定位后的指令及其后续指令后进行地址恢复, 继续分析先前未分析完的代码块. 此外, 为了避免分析陷入循环分支跳转, 且能全面覆盖可能执行的指令, MCBA 在每次跳转前检查当前的指令地址是否存在于返回地址队列中, 如果存在则不进行跳转, 直接继续顺序分析. 需要注意的是, 由于静态分析无法获取运行时的数据状态, 因此无法确定各种分支的判断条件, 所以 MCBA 旨在尽可能按执行逻辑提取操作码序列, 对可能执行的指令进行覆盖, 但不强求对所有可能的执行路径进行覆盖. 当当前指令为函数调用指令时, 首先判断调用指令的操作数是用户函数还是系统函数: 对于用户函数, 则执行与跳转指令相同的分析; 对于系统函数, 则将系统函数名进行保存, 添加到当前操作码序列中. 通过上述逻辑, 可以在保证分析效率的前提下有效地防止分析陷入循环死锁, 覆盖循环分支中的所有有效指令地址. 此外, 由于每次分支分析结束后从返回地址队列中的跳转地址开始继续分析, 不需要对跳转之前的重复前置指令进行重复分析, 避免了程序分析中可能出现的路径爆炸问题.

MCBA 最终输出包含丰富程序上下文语义的操作码序列 (包括部分系统函数名字符串), 并将其存储以作为下一阶段的输入. 与直接从原始二进制流中提取操作码相比, 基于反汇编的执行导向的操作码提取可以以简化、精准的方式将目标程序转换为包含上下文关系的操作码序列文本.

3.2　基于 BERT 滑动窗口的向量嵌入

在向量嵌入阶段, 需要将静态分析阶段获得的具有细粒度程序上下文语义的操作码序列转换为适合神经网络模型输入的形式. 考虑到包含程序语义的操作码序列具有顺序性、上下文依赖性并具有一定的语法结构 (例如函

数调用、条件跳转等固定的操作组合), 这与自然语言句子具有高度的相似性. 与理解自然语言句子逻辑含义时考虑的单词顺序和含义相似, 程序中的操作码也时按照特定顺序排列, 并反映了程序的结构和逻辑. 因此考虑使用能够理解语言上下文和语义的 BERT 自然语言处理模型帮助解析操作码序列, 并从中学习程序的语义特征表示. 由于 BERT 的预训练任务 MLM 和 NSP 都无法直接满足本文的任务需求, MCBA 仅使用 BERT 作为编码器来获取程序操作码字符串的向量表示形式, 并使用 [CLS] 向量作为程序特征向量. MCBA 将静态分析得到的操作码序列组合成一个带空格的长句子, 整体输入到 BERT 进行编码处理得到代表当前样本的 [CLS] 向量, 如图 3 所示.

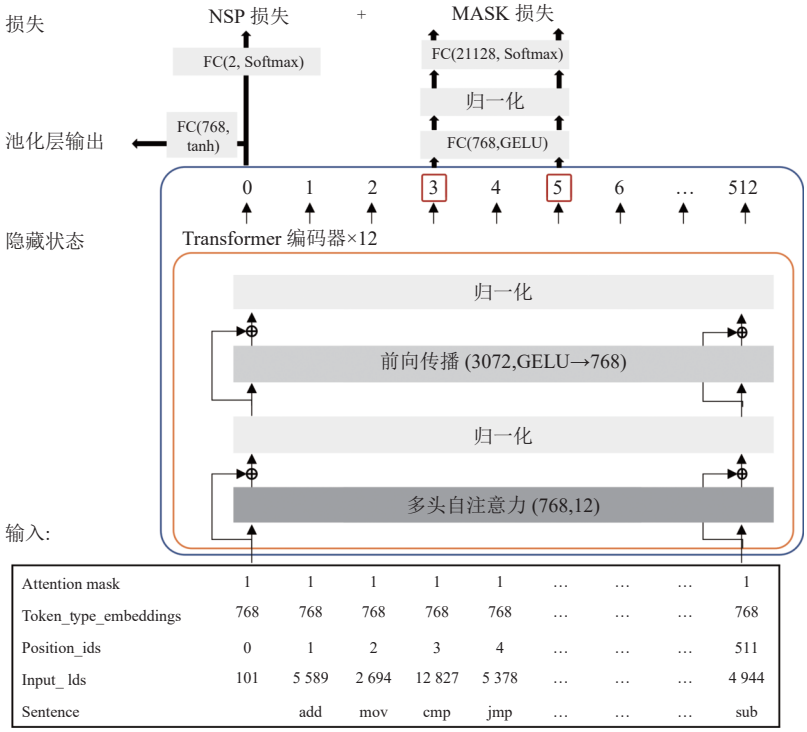


图 3 向量嵌入

操作码助记符本身极为简洁, 与自然语言有所差异. 因此考虑将其转换为更接近自然语言的词语, 例如: mov 转换为 move、jmp 转换为 jump 等. 此外, 考虑到自然语言句子本身不会过长和模型开销的实用性等因素, BERT 模型输入上限限制在 512 个单词. 然而, 虽然恶意软件的体积较小, 但其操作码数量往往超过 512 个. 现有的相关工作[32,33]直接将提取的操作码按顺序截断, 只取小于 512 个操作码进行分析. 虽然这些工作已经被证明是有效的, 但这种方式会丢弃掉大量与分类任务相关的有价值信息. 考虑到 BERT 进行上下文计算生成向量的过程是资源密集型的, 输入数据的大小无法直接扩展. 本文设计了 BERT 滑动窗口的方式进行向量嵌入, 以更多的利用程序指令信息. MCBA 将最大输入形式的 BERT 作为滑动窗口对程序操作码字符串进行处理, 使用 512 个操作码作为一个处理单元, 获得多个 BERT 向量输出. 然后将代表每个输出整体特征的 [CLS] 向量拼接, 生成代表整个程序的特征向量 (如图 1 和图 3 所示). 需要注意的是虽然随着 BERT 滑动窗口数量的增加, 模型可以提取更多特征信息, 但其所需的计算资源也将迅速增长. 经过实验 (第 4.4 节) 评估, MCBA 的 BERT 的窗口数量设置为 3 个.

3.3 基于几何中位数子空间和瓶颈自编码器的模型优化训练和分类

为了提高系统的稳定性, 提高在概念漂移场景下的性能表现, 本节对模型训练过程进行优化, 通过特征映射和降维, 提取与任务目标相关的有效特征. 在自然语言理解领域的相关研究中, 模型在不同数据集上的性能变化一直

是研究热点. 依赖于训练数据集的特点, 学习模型在训练过程中容易学到与数据集本身分布相关而与实际任务无关的虚假特征依赖, 引入虚假的相关性. 因此当模型数据集分布发生变化, 虚假依赖性会导致模型性能降低, 具体表现为在固定的训练样本上性能优秀, 而在分布外的样本上性能不佳.

在本文的研究问题中, 由于概念漂移前后的样本数据分布发生了变化, 因此可以将概念漂移前后的恶意样本数据视为来自两个不同数据集的数据. 鉴于在旧数据上进行训练, 在新数据上进行预测的概念漂移场景与上述自然语言理解中的数据集差异带来的虚假相关性问题类似, 因此可以使用一系列特征降维和提取算法, 提取出与任务目标相关的特征信息, 去除虚假依赖性.

几何中位数子空间常用于特征降维, 可以将高维特征向量映射到低维并反映其实质分布特点. 可以使用几何中位数子空间将 BERT 滑动窗口处理生成的特征向量进行降维提取. 瓶颈自编码器通过解构和重构特征向量, 学习到特征的低维紧凑表示, 捕获输入数据的关键特征. 因此可以利用瓶颈自编码器对进一步对输入特征进行处理, 捕获核心的特征分布. 在模型训练和分类阶段, 使用基于几何中位数子空间和瓶颈自编码器对特征进行降维和提取, 并使用简单的全连接层进行分类预测. 为了确定和优化几何子空间的投影矩阵以及瓶颈自编码器的权重参数, 考虑在损失函数中增加对应的数据项表示. 在训练过程中根据损失反馈对几何中位数子空间的投影矩阵以及瓶颈自编码器的各层网络参数进行调整.

3.3.1 几何中位数子空间

为了减少特征冗余达到去偏效果, MCBA 使用几何中位数子空间的思想进行特征降维投影. 以 BERT 滑动窗口输出的拼接向量为输入, 使用可训练的投影矩阵 **A** 对其进行降维投影, 从而进行初步的特征降维和信息提取. 由于实际情况中, 样本大小不同可能产生不同长度的 [CLS] 拼接向量, 因此为了对齐输入维度, MCBA 对较短向量进行无效位填充, 统一所有样本的输入向量维度为 1 533, 从而对齐 3 个滑动窗口的大小. 然后通过投影矩阵 **A** 将拼接向量降维至与下一阶段自编码器的输入维数相匹配的向量.

几何中位数子空间基于几何中位数的概念和子空间建模, 通过在特征空间中找到一个子空间, 使得该子空间数据集样本点的投影距离之和最小, 从而提取出样本中共享的特征信息, 去除个体差异, 得到更具代表性的特征表示. 几何中位数子空间投影的关键在于求解一个正交投影器使投影后的样本空间满足几何中位数子空间的条件. 假设特征向量为 **h**, 几何中位数子空间的正交投影器为 P, **h** 由 P 投影映射到子空间的向量 **z**, 则计算几何中位数子空间的过程可以用以下数学表达式表示:

$$\min\left(\sum \|\mathbf{z}-P(\mathbf{h})\|_2\right) \tag{1}$$

通过最小化投影的欧氏距离 (即 L2 范数), 可以训练出一个几何中位数子空间投影模型, 用于将输入特征向量映射到共享的几何中位数子空间, 提取共享特征并降低特征冗余. 因此可以构造训练损失函数为如下形式:

$$\mathcal{L}_{\text{projection}}(\mathbf{A}) = \|\mathbf{z}_i - \mathbf{A}(\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{z}_i\|_2 + \|\mathbf{A}\mathbf{A}^{\mathrm{T}} - \mathbf{I}\|_F^2 \tag{2}$$

其中, **A** 代表几何中位数子空间的投影矩阵, $\mathbf{A}^{\mathrm{T}}$ 代表矩阵 **A** 的转置, **I** 代表单位矩阵. 损失函数的第 1 项计算出满足几何中位数子空间的投影矩阵 **A**, 第 2 项将矩阵 **A** 逼近为一个正交矩阵.

3.3.2 瓶颈自编码器

为了进一步对特征中的有效信息进行提取, MCBA 使用一个瓶颈自编码器架构的网络进行进一步特征筛选. 瓶颈自编码器是一种无监督学习模型, 用于学习输入数据的紧凑表示或特征提取. 它由编码器和解码器两部分组成, 通过将输入数据压缩到低维编码空间, 再将其解码重构回原始数据空间, 通过尽可能使重构数据接近输入数据, 学习到最能表达输入特征的低维编码形式. 此外, 为了进一步降维, MCBA 再次使用投影矩阵 **B** 对瓶颈自编码器生成的中间向量进行处理.

编码器 (encoder): 编码器将输入数据映射到低维编码空间. 由一系列的隐藏层组成, 每一层通过非线性的激活函数对输入数据进行变换. 编码器的目标是学习到输入数据的紧凑表示, 并提取出数据的重要特征. 通常, 编码器的隐藏层逐渐减少维度, 最终将数据压缩到编码层, 形成低维的编码表示. MCBA 使用较为常见的 3 层编码器结构, 考虑到来自几何中位数特征子空间的输出层数, MCBA 编码器的第 1 层设置为 768 维. 由于编码器的中间层

维数会影响提取信息的质量, 因此经过实验调优, 最终 MCBA 的 3 层编码器维数确定为 768 维, 356 维, 256 维 (详见第 4.4 节实验).

几何投影矩阵 **B**: 对来自编码器的 256 维中间向量进行处理, 降维至 192 维, 作为分类器的输入向量 **y**.

解码器 (decoder): 解码器接收编码器输出的低维编码, 并将其解码回原始数据空间. 解码器由一系列的隐藏层组成, 通过逆向的变换过程逐渐增加维度, 最终重构出与输入数据尽量相似的重建数据. 解码器的目标是尽可能准确地还原原始数据, 使重建数据与输入数据之间的差异最小化. 解码器通常与编码器结构逆向对应, 因此 MCBA 使用 3 层解码器结构, 每层大小分别设置为 256 维, 356 维, 768 维.

损失函数 (loss function): 瓶颈自编码器的训练过程使用损失函数来衡量重建数据与输入数据之间的差异. 常用的损失函数包括均方差 (mean squared error) 或交叉熵 (cross entropy). 通过最小化损失函数, 瓶颈自编码器的编码器和解码器可以逐渐优化, 使得重建数据尽可能接近原始数据, 从而学习出最有代表性的低维数据表示形式. Vincent 等人[34]研究表明, 自编码器的解码器重构向量 **z** 并不是编码器输入 **x** 的精确重建, 其损失函数可以表示为公式 (3):

$$\mathcal{L}_{\text{recon}} \propto -\log p(\mathbf{x}|\mathbf{z}) \tag{3}$$

因此, 最小化损失函数等价为下面形式:

$$\min(\mathbb{E}[\mathcal{L}_{\text{recon}}(\mathbf{x},\mathbf{z})]) = \max(\mathbb{E}[\log \mathbb{P}(\mathbf{x}|\mathbf{z})]) \tag{4}$$

通过训练使 **z** 不断逼近 **x**, 从而得到有效的低维表达形式 **y** 作为输出, 进行下一阶段的分类预测.

MCBA 使用简单的全连接层进行分类工作. 通过接收来自瓶颈自编码器的输出 **y**, 最终得到对应的类别预测结果.

4 实验分析

4.1 实验数据

为了对分类模型的性能表现进行全面测试, 本文分别在普通环境下和概念漂移环境下进行恶意软件分类实验. 为此, 实验分别使用两个不同的数据集: MalwareBazaar[4]和 MalwareDrift[4], 如表 1 和表 2 所示.

表 1 实验数据集 MalwareBazaar

家族	样本数量
Gozi	767
GuLoader	589
Heodo	214
IcedID	578
Njrat	942
Trickbot	881
Total	3971

表 2 实验数据集 MalwareDrift

家族	漂移前样本	漂移后样本
Bifrose	171	107
Ceeinject	90	458
Obfuscator	143	61
Vbinject	379	653
Vobfus	64	218
Winwebsec	218	269
Zegost	180	114
Total	1245	1 880

MalwareBazaar 数据集: Ma 等人[4]从恶意软件数据网站 MalwareBazaar[35]上选择了近年排名前 6 的恶意软件家族, 分别为每个家族下载了 1000 个最近上传的恶意软件样本. 为了确保数据集的有效性, Ma 等人删除了非 PE 格式的样本, 然后利用 AVClass[36]和 Joe security[37]检查样本的家族标签, 剔除标签不一致的噪声样本. 最后, 筛选出 6 个家族共 3 971 个 PE 恶意软件样本组成数据集 MalwareBazaar.

MalwareDrift 数据集: Wadkar 等人[38]发现代码变更在 χ^2 时间线统计量中表现为尖峰, 该统计量用于衡量在滑动时间窗口上使用 PE 恶意软件特征训练的支持向量机 (SVM) 中的权重差异. 类似地, 恶意软件家族的演化可以被理解为代码变更. Ma 等人[4]使用文献 [38] 中的数据集和相应的 χ^2 时间线统计图, 确定每个恶意软件家族的演化周期, 并将每个家族的样本分为概念漂移前和概念漂移后. 通过以上步骤, 生成了包含 7 个家族共 3 125 个样本的数据集 MalwareDrift, 用于测试时间间隔对分类方法的影响. 漂移前和漂移后的样本分别来自 2015 年之前和

2020 年之后.

4.2 对比工作

为了验证 MCBA 相对其他方法的性能表现, 需要将其与当前最先进工作进行广泛比较实验. 对比工作除了包括基于字节流分析的方法之外还应包括基于图像转换和基于反汇编的方法.

文献 [4] 旨在系统性地研究不同 Windows 恶意软件分类方法的性能, 并对 9 种最先进的基于学习的 Windows 恶意软件家族分类方法进行了实证研究. 其实验结果较为全面地代表了当前 Windows 恶意软件分类工作的研究现状. 这 9 种最先进的方法如下.

• MAGIC[29]是一种基于反汇编的端到端的恶意软件检测方法, 通过提取程序属性函数调用图来对恶意软件进行分类. 该方法能够利用深度图卷积神经网络 (DGCNN)[30]有效地挖掘恶意软件信息.

• Word2Vec+KNN[27]是一项基于反汇编的代表性工作, 通过提取恶意软件操作码并使用 Word2Vec 进行向量嵌入处理. 尽管其使用了词向量嵌入模型, 但并没有使用序列信息进行分类. 它将恶意软件反汇编文件建模为一种恶意软件语言, 提取其中的操作码序列作为恶意软件文档, 使用 Word2Vec 模型生成该文档的计算表示. 此工作选择词移距离作为文档间语义相似度的度量标准, 该度量标准计算将文档 A 中的所有嵌入词汇转移到文档 B 中的所有嵌入词汇所需的代价. 最后结合 KNN 算法, 通过计算文档间的语义距离来进行分类.

• MCSC[25]是一种基于反汇编和图像分类相结合的分析方法. 它提取恶意软件操作码序列, 并基于 SimHash[39]进行编码使它们具有相等的长度. 然后, 它将每个 SimHash 值作为二进制像素, 并将 SimHash 位转换为灰度图像. 它使用修改自 LeNet-5[40]的 CNN 结构对这些灰度图像进行分类训练. 在训练 CNN 分类器时, 使用了多哈希和双线性插值来提高模型的准确性, 并使用主要块选择来减少图像生成时间.

• VGG-16[18]是一种具有高拟合能力的基于图像的方法. 它使用 VGGNet 网络中最流行的、具有大量参数的 VGG-16 架构进行恶意软件分类. VGG-16 是 VGGNet 系列中最受欢迎的网络架构. 标准的 VGG-16 包含 13 个卷积层 (conv)、5 个最大池化层 (pool) 和 3 个全连接层. 所有隐藏层使用修正线性单元 (ReLU) 作为激活函数, 并且使用 Softmax 作为输出层. VGG-16 的优点是其结构简单并且具有较强的拟合能力.

• ResNet-50[18]是一种基于图像的恶意软件分类方法. 该方法使用 ResNet 系列中的一个典型网络 ResNet-50. ResNet-50 是一个包含 50 个隐藏层的深层网络, 使用全局平均池化和一个 Softmax 作为最终的预测层. 与 VGG-16 网络相比, ResNet-50 能够以更少的参数, 高效且较完整地捕捉恶意软件信息.

• Inception-V3[18]是一个将 Inception-V3 应用于基于图像的恶意软件分类的工作. 它将二进制文件转换为图像并使用新型的 Inception-V3 架构卷积神经网络进行分类. Inception-V3 包含对称和非对称的结构, 包括卷积 (conv)、平均池化 (average pool)、最大池化 (max pool)、连接 (concatenation)、dropout 和全连接层 (FCs). 与 ResNet-50 类似, Inception-V3 最终也使用全局平均池化, 然后使用 Softmax 作为最终的预测层. 与 VGG-16 相比, Inception-V3 更有效, 并且其参数数量比 ResNet-50 更少.

• IMCFN[18]是使用微调卷积神经网络进行基于图像的恶意软件分类工作的代表. 它使用了 VGG-16 的变体, 将前两个全连接层的神经元个数从 4 096 减少到 2 048, 并添加了 dropout 层减少过拟合所带来的负面影响.

• CBOW+MLP[8]是一种基于字节流分析的恶意软件家族分类方法, 它结合了 Word2Vec[23]和多层感知器 (MLP). 其关键思想是同一家族样本中的字节之间存在相似关系, 并且与不同家族的样本明显不同. 因此, 原始字节的向量矩阵是恶意软件分类的有效特征. 该方法首先对原始二进制数据进行预处理, 去除 5 个或更多连续的 0x00 或 0xCC (无意义的字节). 然后将每个文件作为一个语料库, 认为它由从 0x00 到 0xFF 的 256 个字节组成, 然后使用 Word2Vec 中的连续词袋模型 (CBOW) 获取文件中 256 个字节的嵌入向量, 并将每个文件表示为一个字节向量升序矩阵. MLP 将这些矩阵作为输入, 输出对应的家族类别.

• MalConv[9]是一个端到端的基于字节流分析的恶意软件分析模型. 该方法首先使用嵌入层将原始字节映射到一个固定的 8 维向量. 通过考虑局部和全局上下文, 它可以捕捉原始二进制中的上层位置不变性. 然后, MalConv 使用具有较大步幅的浅层卷积神经网络 (CNN) 通过数据并行的方式更好地平衡了计算工作负载, 缓解了第 1 层

卷积的 GPU 内存消耗问题. 作为一个浅层 CNN 架构, MalConv 解决了读取整个恶意软件字节流时内存的消耗限制, 捕捉了原始二进制中的全局位置不变性并且允许嵌入层与卷积层联合训练, 以获得更好的特征提取效果.

4.3 评价指标与参数设置

参照文献 [4] 对 9 个代表性工作的实验设置, 本文使用 10 倍交叉验证和准确率 (*A*)、精确率 (*P*)、召回率 (*R*) 和 *F*1 分数 (*F*1) 这 4 个评价指标对 MCBA 及相关对比工作进行实验评估.

- 真阳性 (*TP*): 模型将属于某个类别的样本正确分类为该类别的次数.
- 真阴性 (*TN*): 模型将不属于某个类别的样本正确分类为非该类别的次数.
- 假阳性 (*FP*): 模型将不属于某个类别的样本错误分类为该类别的次数.
- 假阴性 (*FN*): 模型将属于某个类别的样本错误分类为非该类别的次数.
- 准确率=(*TP*+*TN*)/(*TP*+*TN*+*FP*+*FN*).
- 召回率=*TP*/(*TP*+*FN*).
- 精准率=*TP*/(*TP*+*FP*).
- *F*1=2×精准率×召回率/(精准率+召回率).

实验中各工作的参数设置如后文表 3 所示.

4.4 消融实验

为了验证滑动 BERT 窗口以及基于几何中位数子空间和自编码器优化方法在信息提取和特征降维上的有效性, 需要对 MCBA 在同一数据集上进行启用优化和禁用优化的对比实验. 为了实验的全面性, 本文分别在代表普通场景的 MalwareBazaar 数据集和代表概念漂移场景的 MalwareDrift 数据集上进行评估.

首先验证 BERT 滑动窗口的有效性. 在 MalwareBazaar 和 MalwareDrift 数据集上, 我们分别开启不同数量的滑动窗口, 并观察不同数量的滑动窗口对分类准确率的影响. 然后我们验证基于几何中位数子空间和自编码器优化方法的有效性. 分别在两个数据集上禁用和启用不同维数的编码器进行分类性能评估. 实验结果如图 4 所示.

表 3 实验对比工作参数设置

实验工作	优化器	学习率	Batch size
ResNet-50	Adam	1E–3	64
VGG-16	SGD	5E–6	64
Inception-V3	Adam	1E–3	64
IMCFN	SGD	5E–6	32
MAGIC	SGD	1E–4	10
Word2Vec+KNN	—	—	—
MCSC	SGD	5E–3	64
CBOW+MLP	SGD	1E–3	128
MalConv	Adam	1E–3	32
MCBA	Adam	5E–5	12

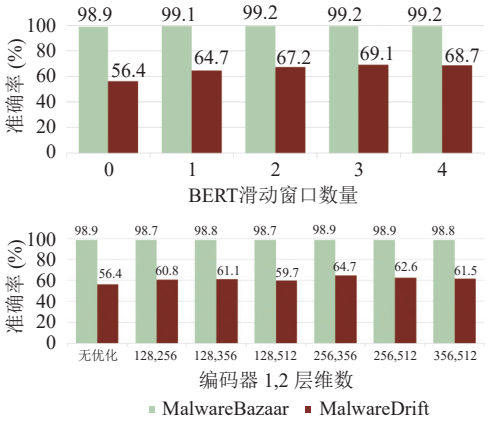


图 4 消融实验

实验结果表明, 无论是在普通场景还是概念漂移下, 启用 BERT 滑动窗口后, 分类准确率都有所提高. 这是因为启用滑动窗口可以使学习模型接受更多来自原始文件的序列信息, 从而获得具有更大维度和更丰富信息的程序特征向量表示. 随着滑动窗口增多, 可以观察到分类准确率也得到提高. 当滑动窗口数量大于 3 时, 分类准确率几乎不再变化. 这意味着此时滑动窗口数量与分类准确率之间出现明显的边际效应递减, 几乎不再带来性能改善. 考虑到实际应用和资源消耗, MCBA 的滑动窗口数量将设置为 3, 并进行后续实验. 在基于几何中位数子空间和自编码器优化方法的评估方面, 对于 MalwareBazaar 数据集, 开启优化的效果并不明显. 这是因为在相同的数据集分布

下, 分类器学到的特征向量中的虚假相关性对当前分析是有益的. 因此, 即使重新构建特征向量以消除虚假相关性, 也很难获得更好的效果. 而在 MalwareDrift 数据集上, 由于训练数据和预测数据的分布不同, 虚假相关性对分类器的负面影响得到明显体现. 因此, 启用基于几何中位数子空间和自编码器优化对虚假相关性进行消除可以使模型取得更好的分类性能. 此外, 自编码器的维数设置也会对性能产生影响. 编码器中间层维数的大小决定了信息提取的质量. 过小的维数可能导致信息压缩过量, 从而发生信息丢失. 过大的维数可能导致信息提取能力不足, 无法去除冗余信息. 通过实验结果可以看到, 当维数设置为 256 维, 356 维, 768 维 (固定) 时, 可以达到较好的分类表现.

4.5 普通场景下的分类性能实验

为了验证 MCBA 的有效性, 需要先在普通场景下进行分类性能实验. 通过与 9 个最先进的相关工作在数据集 MalwareBzaaar 上进行同条件对比实验, 得到实验结果如表 4 所示.

表 4　MCBA 与 9 种先进相关工作性能对比实验 (%)

类别	方法	分类性能			
		准确率	精准率	召回率	*F*1值
基于图像转换	ResNet-50	96.68	96.91	96.75	96.83
	VGG-16	96.35	96.58	96.54	96.56
	Inception-V3	95.83	95.67	95.79	95.73
	IMCFN	97.38	97.53	97.41	97.47
基于反汇编分析	MAGIC	92.82	88.03	87.36	87.45
	Word2Vec+KNN	95.64	93.34	94.29	93.79
	MCSC	96.80	94.97	94.51	94.70
基于字节流分析	CBOW+MLP	97.81	97.92	98.08	98.00
	MalConv	95.92	96.04	96.43	96.20
	MCBA	99.46	99.56	99.43	99.49

从实验结果可以看出, 在普通场景下, MCBA 的性能优于其他 9 种对比方法 1.5%–5.7%. 具体而言, 在基于图像转换的方法中, IMCFN 在 *F*1 性能方面表现最好, 可以达到 97.47%. 基于反汇编分析的方法表现普遍差于其他类别的方法 3%–11%. 基于字节流分析的方法中 CBOW+MLP 的 *F*1 达到了 98%, 它也是 9 个最先进工作中表现最好的. 总体而言, 基于字节流的方法比基于图像和基于反汇编的方法表现更好. 这可能是因为基于字节流的方法能够更好地捕捉程序的上下文语义. 基于反汇编的方法高度依赖于人工特征选择的优劣, 而在不同的分析场景下, 人工选择的特征并不总是适用, 因此分类性能会受到分析场景的影响. 此外, 基于反汇编的方法还受限于静态分析工具的分析策略和能力. 基于图像的方法不需要专家知识进行反汇编和手动特征选择, 但将恶意样本转换为图像可能会引入新的超参数. 例如, 使用 CNN 的方法通常将图片的宽度限制为 224, 这要求将恶意样本的二进制文件流以 CNN 输入尺寸为宽度进行重新调整, 这在字节流之间添加了可能不存在的上下空间特征, 从而对分类产生负面影响. 基于字节流分析的方法性能表现都较好, 其中 CBOW+MLP 的性能与 MCBA 最为接近, 这是因为它也采用了使用学习模型理解程序语义的思想. 但由于其固定的特征数组和缺乏对上下文信息的挖掘, 在面对概念漂移场景时, 其性能非常不稳定. 我们将在第 4.6 节中讨论这个问题.

总的来说, MCBA 在普通场景下表现优秀, 证明了 MCBA 在一般情况下的实用性. 这主要得益于 MCBA 能够以行为为导向提取程序的深层上下文语义信息并进行转化优化.

4.6 概念漂移场景下的分类性能实验

在实际应用中分类模型时常会需要在概念漂移的场景中进行工作, 因此方法适应概念漂移场景的能力决定了其处理新恶意样本的能力和实际应用的价值. 为了评估在概念漂移场景下, MCBA 与其他最先进相关工作的性能, 我们在 MalwareDrift 数据集上进行了对比实验.

为了模拟概念漂移场景, 实验将 MalwareDrift 的漂移前数据设置为训练集, 模拟较为旧的训练场景, 将 MalwareDrift 的漂移后数据作为实验的测试集, 模拟在时间发生推移后现实世界中出现的新样本, 进行分类预测测试.

实验结果如表 5 所示, 特别地, 实验中选择 IMCFN 作为基于图像转换的代表方法, 因为它在之前的实验中表现最佳. 从结果表中可以看出, 在概念漂移场景下, 所有方法都有不同程度的性能下降, 其中 CBOW+MLP 的下降幅度最大, *F*1 值仅有 10.88%. 在概念漂移场景下, Word2Vec+KNN 的性能表现在 9 种对比工作中最好, *F*1 值为 43.87%. 其他方法无论基于图像转换基于字节流分析, 在概念漂移场景下的性能下降都较大. 这是因为大部分基于图像转换和字节流分析的方法都直接将整个二进制文件作为输入进行处理. 随着时间的推移, 同一恶意软件家族的文件结构可能发生显著变化, 这无疑会显著改变二进制文件的特征, 影响这些分析方法的正常判断. 对于基于反汇编的方法, 由于同一家族的恶意软件行为逻辑可能不会发生明显变化, 因此反汇编提取的分析特征可能仍然有效. 但这极其依赖于定义和抽取的反汇编特征的有效性, 例如同属于基于反汇编分析的 MAGIC 和 MCSC, 其性能表现相差 15.52%.

表 5　概念漂移场景下各方法性能表现 (%)

类别	方法	分类性能			
		准确率	精准率	召回率	*F*1值
基于图像转换	IMCFN	49.90	52.74	44.42	42.10
基于反汇编分析	MAGIC	42.15	34.69	33.07	28.30
	Word2Vec+KNN	49.18	48.73	51.80	43.87
	MCSC	50.58	46.97	48.25	43.82
基于字节流分析	CBOW+MLP	17.44	11.52	14.45	10.88
	MalConv	46.50	39.19	39.49	35.51
	MCBA	68.72	68.20	58.47	54.59

在概念漂移场景中, MCBA 的性能表现优于其他方法, 其 *F*1 值为 54.59%. MCBA 之所以对新样本具有更强的泛化能力, 是因为采用了有效的特征提取和处理方法并对程序的上下文语义拥有深刻理解. 与一般的基于图像转换和基于字节流分析的方法不同, MCBA 不以整体二进制文件作为输入, 而是先对恶意软件原始字节流进行反汇编分析, 剔除冗余信息, 只保持对行为相关的操作码序列的关注, 这可以避免引入文件结构信息等方面的额外噪声. 此外与一般的反汇编分析的方法不同, MCBA 使用深度学习语言模型理解程序的上下文语义, 具有高度的灵活性和适应性, 不依赖固定的反汇编特征提取模式, 可以依靠神经网络理解难以人工获取的深层程序上下文语义. 除此之外, MCBA 只保持对任务相关特征的关注, 使用关键特征进行分类. 这些都对 MCBA 恶意软件分类性能表现稳定性的提高产生了积极作用.

4.7　运行开销对比

在实际应用中, 模型的运行开销使衡量一个模型实际可用性的重要指标. 本节对 MCBA 与 9 个对比工作在 MalwareBazaar 数据集上的模型训练时间、样本预处理时间和单样本预测时间 (如表 6 所示) 进行评估对比, 以验证 MCBA 的实用性.

表 6　MCBA 与 9 种先进相关工作运行开销

类别	方法	运行时间		
		模型训练 (min)	预处理时间 (s/个)	预测时间 (ms/个)
基于图像转换	ResNet-50	8.4	0.7	2.6
	VGG-16	44.0	0.7	2.2
	Inception-V3	6.4	0.7	2.0
	IMCFN	18.8	0.7	2.2
基于反汇编分析	MAGIC	246.0	17.8	23.6
	Word2Vec+KNN	<0.1	17.7	95 243.3
	MCSC	1.1	4.5	3 477.8
基于字节流分析	CBOW+MLP	0.8	1.1	5 441.0
	MalConv	65.4	0.3	14.0
	MCBA	2.8	1.0	1.8

在模型训练阶段, 运行开销会根据分类方法类型和所使用的学习算法而表现出较大差异. 从结果可以看出, 基于图像转换的方法 ResNet-50、VGG-16、Inception-V3 和 IMCFN 分别需要 8.4 min、44.0 min、6.4 min 和 18.8 min 进行模型训练. MAGIC 需要高达 246 min 进行模型训练, 是模型训练时间最长的方法. 这是因为这些方法在训练过程中使用了复杂的卷积神经网络模型, 需要大量的计算资源和开销对复杂的模型参数进行调整. Word2Vec+KNN 和 CBOW+MLP 的训练时间最短, 因为它们使用简单的词嵌入模型和机器学习分类算法. 然而, 简单的词向量模型缺乏对上下文信息的关注从而影响分类性能的稳定性. MCBA 的基于几何中位数子空间和瓶颈自编码器的模型训练仅需要 2.8 min 即可充分理解输入样本的上下文信息和模型调整, 能够以较短的训练时间达到优秀的分类性能.

在样本预处理阶段, 基于图像转换的方法直接将二进制文件转换并重构为灰度图像, 不需要其他的分析处理, 因此其处理速度很快. 基于反汇编分析的方法通常耗时较长, 分别需要 17.8 s、17.7 s 和 4.5 s. 这是因为自定义的反汇编分析过程会带来额外的分析开销. MCBA 使用一个仅关注操作码的轻量级反汇编分析模块和预训练的 BERT 模型进行特征向量生成, 平均仅需要 1.0 s 就可以完成一个样本的预处理. 这其中包括 0.9 s 的反汇编处理和小于 0.1 s 的 BERT 词向量嵌入过程.

在单样本预测阶段, 基于图像转换的方法表现都十分优秀, 预测单个样本的时间仅需要 2 ms. Word2Vec+KNN 和 CBOW+MLP 的预测时间非常长, 分别需要 95 243 ms 和 5 441 ms. 这可能是因为它们的特征处理方式过于简单, 分类方法不够有效, 因此需要更长的时间生成预测结果. MCBA 的单样本预测时间为 1.8 ms, 快于所有的方法, 这得益于有效的特征降维对特征向量进行的简化.

总之, MCBA 在常见场景和概念漂移场景下的分类性能均优于其他 9 种最先进的方法并且拥有较低的运行开销, 这证明了 MCBA 的有效性以及较高的实际应用意义.

5 总 结

针对概念漂移场景下恶意软件分类模型的性能极易受到影响而产生性能下降的问题. 本文提出了一种基于 BERT 和自编码器的概念漂移恶意软件分类优化方法. 通过将执行导向的程序操作码序列作为自然语言输入到 BERT 中, 对程序隐含的深层上下文信息进行捕获. 进一步通过结合几何中位数子空间投影和瓶颈自编码器的方法进行特征降维和提取, 获得与任务相关的特征信息, 以此增强分类性能的稳定性, 减少概念漂移为分类模型带来的负面影响. 通过与 9 种最先进的工作进行对比实验, 验证了本文所提出的方法无论是否在概念漂移场景下, 都能表现出比现有 Windows 恶意软件分类相关工作更优秀的分类准确性和鲁棒性.

但是, 目前工作仍具有一些局限性, 例如基于静态分析的方法容易受到来自攻击者各种攻击手段影响[41], 因此在未来工作中, 我们将会尝试更多特征信息提取方法, 增强对模型训练的优化. 此外还将对方法的预处理部分进行改善, 增加自动化查壳和自动化脱壳的预处理模块, 以面对更复杂的实际应用情况.

References:

[1] Gibert D, Mateu C, Planes J. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. Journal of Network and Computer Applications, 2020, 153: 102526. [doi: 10.1016/j.jnca.2019.102526]

[2] Shalaginov A, Banin S, Dehghantanha A, Franke K. Machine learning aided static malware analysis: A survey and tutorial. arXiv: 1808.01201, 2018.

[3] Wang JL, Zhang C, Qi XY, Rong Y. A survey of intelligent malware detection on windows platform. Journal of Computer Research and Development, 2021, 58(5): 977–994 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2021.20200964]

[4] Ma YX, Liu S, Jiang JJ, Chen GH, Li KQ. A comprehensive study on learning-based PE malware family classification methods. In: Proc. of the 29th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. Athens: ACM, 2021. 1314–1325. [doi: 10.1145/3468264.3473925]

[5] Hassen M, Chan PK. Scalable function call graph-based malware classification. In: Proc. of the 7th ACM on Conf. on Data and Application Security and Privacy. Scottsdale: ACM, 2017. 239–248. [doi: 10.1145/3029806.3029824]

[6] Kinable J, Kostakis O. Malware classification based on call graph clustering. Journal in Computer Virology, 2011, 7(4): 233–245. [doi:

10.1007/s11416-011-0151-y]

[7] Kong DG, Yan GH. Discriminant malware distance learning on structural information for automated malware classification. In: Proc. of the 19th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Chicago: ACM, 2013. 1357–1365. [doi: 10.1145/2487575.2488219]

[8] Qiao YC, Zhang B, Zhang WZ. Malware classification method based on word vector of bytes and multilayer perception. In: Proc. of the 2020 IEEE Int'l Conf. on Communications. Dublin: IEEE, 2020. 1–6. [doi: 10.1109/ICC40277.2020.9149143]

[9] Raff E, Barker J, Sylvester J, Brandon R, Catanzaro B, Nicholas CK. Malware detection by eating a whole EXE. In: Proc. of the Workshops of the 32nd AAAI Conf. on Artificial Intelligence. New Orleans: AAAI, 2018. 268–276.

[10] Arjovsky M, Bottou L, Gulrajani I, Lopez-Paz D. Invariant risk minimization. arXiv:1907.02893, 2020.

[11] Marcus G. Deep learning: A critical appraisal. arXiv:1801.00631, 2018.

[12] Nataraj L, Karthikeyan S, Jacob G, Manjunath BS. Malware images: Visualization and automatic classification. In: Proc. of the 8th Int'l Symp. on Visualization for Cyber Security. Pittsburgh: ACM, 2011. 4. [doi: 10.1145/2016904.2016908]

[13] Kancherla K, Mukkamala S. Image visualization based malware detection. In: Proc. of the 2013 IEEE Symp. on Computational Intelligence in Cyber Security. Singapore: IEEE, 2013. 40–44. [doi: 10.1109/CICYBS.2013.6597204]

[14] Gibert D, Mateu C, Planes J, Vicens R. Using convolutional neural networks for classification of malware represented as images. Journal of Computer Virology and Hacking Techniques, 2019, 15(1): 15–28. [doi: 10.1007/s11416-018-0323-0]

[15] Jain M, Andreopoulos W, Stamp M. Convolutional neural networks and extreme learning machines for malware classification. Journal of Computer Virology and Hacking Techniques, 2020, 16(3): 229–244. [doi: 10.1007/s11416-020-00354-y]

[16] Jang JW, Woo J, Yun J, Kim HK. Mal-netminer: Malware classification based on social network analysis of call graph. In: Proc. of the 23rd Int'l Conf. on World Wide Web. Seoul: ACM, 2014. 731–734. [doi: 10.1145/2567948.2579364]

[17] Kalash M, Rochan M, Mohammed N, Bruce NDB, Wang Y, Iqbal F. Malware classification with deep convolutional neural networks. In: Proc. of the 9th IFIP Int'l Conf. on New Technologies, Mobility and Security. Paris: IEEE, 2018. 1–5. [doi: 10.1109/NTMS.2018.8328749]

[18] Vasan D, Alazab M, Wassan S, Naeem H, Safaei B, Zheng Q. IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. Computer Networks, 2020, 171: 107138. [doi: 10.1016/j.comnet.2020.107138]

[19] Vasan D, Alazab M, Wassan S, Safaei B, Zheng Q. Image-based malware classification using ensemble of CNN architectures (IMCEC). Computers & Security, 2020, 92: 101748. [doi: 10.1016/j.cose.2020.101748]

[20] Moskovitch R, Stopel D, Feher C, Nissim N, Elovici Y. Unknown malcode detection via text categorization and the imbalance problem. In: Proc. of the 2008 IEEE Int'l Conf. on Intelligence and Security Informatics. Taipei: IEEE, 2008. 156–161. [doi: 10.1109/ISI.2008.4565046]

[21] Jain S, Meena YK. Byte level *n*-gram analysis for malware detection. In: Proc. of the 5th Int'l Conf. on Information Processing. Bangalore: Springer, 2011. 51–59. [doi: 10.1007/978-3-642-22786-8_6]

[22] Zhang FY, Zhao TZ. Malware detection and classification based on *n*-grams attribute similarity. In: Proc. of the 2017 IEEE Int'l Conf. on Computational Science and Engineering (CSE) and IEEE Int'l Conf. on Embedded and Ubiquitous Computing. Guangzhou: IEEE, 2017. 793–796. [doi: 10.1109/CSE-EUC.2017.157]

[23] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. In: Proc. of the 1st Int'l Conf. on Learning Representations. Scottsdale: ICLR, 2013.

[24] Ahmadi M, Ulyanov D, Semenov S, Trofimov M, Giacinto G. Novel feature extraction, selection and fusion for effective malware family classification. In: Proc. of the 6th ACM Conf. on Data and Application Security and Privacy. New Orleans: ACM, 2016. 183–194. [doi: 10.1145/2857705.2857713]

[25] Ni S, Qian Q, Zhang R. Malware identification using visualization images and deep learning. Computers & Security, 2018, 77: 871–885. [doi: 10.1016/j.cose.2018.04.005]

[26] Sun GS, Qian Q. Deep learning and visualization for identifying malware families. IEEE Trans. on Dependable and Secure Computing, 2021, 18(1): 283–295. [doi: 10.1109/TDSC.2018.2884928]

[27] Awad Y, Nassar M, Safa H. Modeling malware as a language. In: Proc. of the 2018 IEEE Int'l Conf. on Communications. Kansas City: IEEE, 2018. 1–6. [doi: 10.1109/ICC.2018.8422083]

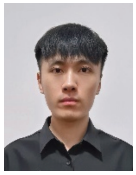
[28] Kusner MJ, Sun Y, Kolkin NI, Weinberger KQ. From word embeddings to document distances. In: Proc. of the 32nd Int'l Conf. on Machine Learning. Lille: JMLR.org, 2015. 957–966.

[29] Yan JQ, Yan GH, Jin D. Classifying malware represented as control flow graphs using deep graph convolutional neural network. In: Proc. of the 49th Annual IEEE/IFIP Int'l Conf. on Dependable Systems and Networks. Portland: IEEE, 2019. 52–63. [doi: 10.1109/DSN.2019.

00020]
[30] Zhang MH, Cui ZC, Neumann M, Chen YX. An end-to-end deep learning architecture for graph classification. In: Proc. of the 32nd AAAI Conf. on Artificial Intelligence, the 30th Innovative Applications of Artificial Intelligence Conf. and the 8th AAAI Symp. on Educational Advances in Artificial Intelligence. New Orleans: AAAI Press, 2018. 4438–4445.
[31] Jordaney R, Sharad K, Dash SK, Wang Z, Papini D, Nouretdinov I, Cavallaro L. Transcend: Detecting concept drift in malware classification models. In: Proc. of the 26th USENIX Security Symp. Vancouver: USENIX Association, 2017. 625–642.
[32] Alvares J, di Troia F. BERT for malware classification. In: Stamp M, Visaggio CA, Mercaldo F, Di Troia F, eds. Artificial Intelligence for Cybersecurity. Cham: Springer, 2022. 161–181. [doi: 10.1007/978-3-030-97087-1_7]
[33] Oak R, Du M, Yan D, Takawale H, Amit I. Malware detection on highly imbalanced data through sequence modeling. In: Proc. of the 12th ACM Workshop on Artificial Intelligence and Security. London: ACM, 2019. 37–48. [doi: 10.1145/3338501.3357374]
[34] Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol PA. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. The Journal of Machine Learning Research, 2010, 11: 3371–3408.
[35] MalwareBazaar Homepage. 2021. https://bazaar.abuse.ch/
[36] Sebastián M, Rivera R, Kotzias P, Caballero J. AVClass: A tool for massive malware labeling. In: Proc. of the 19th Int'l Symp. on Research in Attacks, Intrusions, and Defenses. Paris: Springer, 2016. 230–253. [doi: 10.1007/978-3-319-45719-2_11]
[37] Joe security LLC. Joe security. 2022. https://www.joesecurity.org/
[38] Wadkar M, Di Troia F, Stamp M. Detecting malware evolution using support vector machines. Expert Systems with Applications, 2020, 143: 113022. [doi: 10.1016/j.eswa.2019.113022]
[39] Manku GS, Jain A, Das Sarma A. Detecting near-duplicates for Web crawling. In: Proc. of the 16th Int'l Conf. on World Wide Web. Banff Alberta: ACM, 2007. 141–150. [doi: 10.1145/1242572.1242592]
[40] LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1989, 1(4): 541–551. [doi: 10.1162/neco.1989.1.4.541]
[41] Ji TT, Fang BX, Cui X, Wang ZR, Liao P, Song SY. Framework for understanding intention-unbreakable malware. Science China Information Sciences, 2023, 66(4): 142104. [doi: 10.1007/s11432-021-3567-y]

附中文参考文献:

[3] 汪嘉来, 张超, 戚旭衍, 荣易. Windows 平台恶意软件智能检测综述. 计算机研究与发展, 2021, 58(5): 977–994. [doi: 10.7544/issn1000-1239.2021.20200964]



赵浩钧(1997－), 男, 博士, 主要研究领域为恶意软件分析, 二进制代码相似性分析.

吴月明(1993－), 男, 博士, CCF 学生会员, 主要研究领域为移动安全, 软件供应链安全, 人工智能安全, 恶意软件分析, 漏洞分析与克隆代码审计.

邹德清(1975－), 男, 教授, 博士生导师, CCF 高级会员, 主要研究领域为云计算安全, 网络攻防与漏洞检测, 软件定义安全与主动防御, 大数据安全与人工智能安全, 容错计算.

金海(1966－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为计算机系统结构, 虚拟化技术, 集群计算, 网格计算, 并行与分布式计算, 对等计算普适计算, 语义网, 存储与安全.

薛文杰(1999－), 男, 硕士, CCF 学生会员, 主要研究领域为软件侧信道漏洞检测, 克隆漏洞检测.