基于深度学习的恶意代码检测综述

严 沛[1,2], 谭舜泉[1,2], 黄继武[1,2]

[1]深圳大学“智能信息处理”广东省重点实验室 深圳 中国 518060
[2]深圳大学“媒体信息内容安全”深圳市重点实验室 深圳 中国 518060

摘要 恶意代码是计算机和网络安全最大的隐患之一。尽管已经有众多检测方法和工具, 但在恶意代码变体快速迭代、代码样本爆炸性增长的形势下, 如何提升恶意代码检测方法的性能, 仍然是当前网络安全领域富有挑战性的热点研究问题。随着人工智能技术的发展, 基于深度学习的恶意代码检测方法逐步引起研究人员的重视。通过使用大量的神经元拟合数据的特征, 可以实现更强的检测性能。相较于早期方法和传统机器学习方法, 基于深度学习的方法能自动提取代码特征, 支持持续学习, 因而逐渐成为恶意代码检测的主流。本文从五个方面对这一主题的研究现状进行回顾和分析: 1)基于熵信息的方法; 2)基于图的方法; 3)基于计算机视觉的方法; 4)基于自然语言处理的方法; 5)基于多维度特征融合的方法。不同于以往的综述工作, 本文一方面根据检测模型的特点进行分类, 并对每个类别的分析方法和架构进行总结; 另一方面通过比较自然图像和文本与恶意代码特征的异同, 试图对恶意代码特征提取的未来改进有所启发。此外, 考虑到生成与对抗技术具有双面性, 且能够促进检测模型的改进与性能提升, 本文对恶意代码检测中的攻防对抗技术进行回顾与分析。目前, 恶意代码检测技术仍存在泛化性与鲁棒性弱、数据集不平衡和概念漂移现象严重等问题, 这些将是未来基于深度学习的恶意代码检测技术研究的主要问题。本文有助于研究人员了解恶意代码检测方法的基本原理、技术方法、现阶段难题与挑战、未来发展方向, 为现有方法的进一步研究和改进提供帮助。

关键词 恶意代码检测; 计算机安全; 网络安全; 深度学习
中图法分类号 TP309.5 DOI 号 10.19363/J.cnki.cn10-1380/tn.2025.05.07

A Survey: Malware Detection Based on Deep Learning

YAN Pei[1,2], TAN Shunquan[1,2], HUANG Jiwu[1,2]

[1] Guangdong Key Laboratory of Intelligent Information Processing, Shenzhen University, Shenzhen 518060, China
[2] Key Laboratory of Media Security, Shenzhen University, Shenzhen 518060, China

Abstract Malware is one of the greatest threats to computer and network security. Despite the availability of numerous detection methods and tools, under the circumstances of the expanding attack scope, rapid iteration of malware variants, and explosive growth of code samples, how to enhance the performance of malware detection methods remains a challenging and hot research topic in the field of network security. With the development of artificial intelligence technology, deep learning-based methods have gradually attracted the attention of researchers, utilizing a large number of neurons to fit data features and achieve stronger detection performance. Compared to early methods and traditional machine learning methods, deep learning-based methods can automatically extract data features and support continuous learning, thus gradually becoming mainstream in malware detection. This paper reviews and analyzes the recent work on this topic from 5 perspectives: 1) Entropy information-based methods; 2) Graph-based methods; 3) Computer vision-based methods; 4) Natural language processing-based methods; 5) Multi-dimensional feature fusion-based methods. Different from previous review works, on the one hand, this paper classifies detection models in the light of their characteristics, and summarizes the analysis methods and architectures of each category; on the other hand, it attempts to inspire future improvements in malware feature extraction by comparing the similarities and differences among natural image and text features with malware features. In addition, considering that generative and adversarial techniques have two sides, which can promote the detection effects and performance improvements, this paper reviews and analyzes the offensive and defensive adversarial techniques in malware detection. Currently, malware detection technology still faces issues such as weak generalization and robustness, imbalanced datasets, and severe concept drift. These will be the main problems for future research on deep learning-based malware detection technology. The comprehensive review of deep learning-based malware detection methods helps researchers understand the basic principles, technical methods, current challenges, and future development directions of malware detection methods, which is conducive to further research and innovation of existing methods.

Key words malware detection; computer security; network security; deep learning

通讯作者: 黄继武, 博士, 教授, Email: jwhuang@szu.edu.cn。
本课题得到国家重点研发计划(No. 2020YFB1805400)资助。
收稿日期: 2023-03-06; 修改日期: 2023-09-01; 定稿日期: 2025-03-04

1 引言

互联网应用的高速发展, 在给普罗大众带来巨大红利的同时, 许多安全问题, 如个人信息泄露、垃圾邮件骚扰、恶意代码入侵等亦随之而来。其中, 恶意代码是网络与计算机安全重要的威胁之一。每年都有许多国家地区、组织和个人因恶意代码入侵而蒙受巨大损失。目前, 主要的恶意代码形式包括[1-4]:

病毒(Virus): 能够附着在其他文件上并通过复制自身进行传播, 当用户执行被感染的文件时, 病毒便会激活并感染其他文件。

蠕虫(Worms): 能够在短时间内大量复制和传播, 并且可以通过多种类型程序进行传播。

根套件(Rootkits): 用于隐藏其自身或其他恶意活动, 使其在计算机系统或网络中难以被检测。

后门(Backdoors): 让攻击者绕过大量甚至所有的认证流程, 实现远程控制计算机, 并在远程系统上运行命令。

特伊洛木马(Trojan Horse): 通常伪装成合法或有诱惑性的软件、文件或链接, 诱使用户主动下载或执行。一旦特伊洛木马进入用户的计算机系统, 它将执行预设的恶意操作。这可能导致数据泄露、系统损坏或其他危害。

勒索软件(Ransomware): 对关键文件进行加密, 迫使用户支付赎金以解密文件。

灰度软件(Grayware): 在功能和性质上介于正常软件与恶意软件之间的软件。虽然灰度软件不一定具有明显的恶意功能, 但其行为可能令人不悦、侵犯隐私或影响系统性能。

广告插件(Adware): 通常在用户不知情的情况下安装在计算机上, 通过在屏幕投放广告以获取收益。这些广告会影响用户的体验, 并在一定程度上影响计算机性能。

间谍软件(Spyware): 在用户并不知情也没有授权的情况下, 非法安装在用户电脑上监视用户在线行为的软件, 该软件可以在后台执行一系列恶意操作, 比如将监视信息发送给攻击者, 从而窃取用户的个人信息和敏感数据。

下载器(Downloader): 该类恶意程序在得到下载的权限后, 会自动下载其他的恶意程序。

在个人层面, 恶意代码会窃取用户的敏感信息, 导致用户隐私泄露和经济损失; 在企业层面, 恶意代码会对企业的关键数据和设施造成损害, 进而影响企业的正常运行; 在国家层面, 恶意代码会窃取保密数据和损坏关键基础设施, 威胁国家安全。在此现状下, 恶意代码检测方法能够在很大程度上减少恶意代码所带来的损失, 这对于保护用户隐私和数据安全、保障企业长期稳定发展、筑牢国家网络空间安全防线而言, 有重要的意义。

尽管目前已经有不少的检测方法, 但随着时间的不断推移, 许多新的问题也逐渐浮现。恶意代码设计者利用加密、混淆、变种等手段提升恶意代码的复杂性和多样性, 使得其难以被检测。传统的基于特征匹配和行为分析的检测方法在面对这些复杂多样的恶意代码时, 往往表现出较低的检测准确率和实时性。在大数据时代下, 随着数据量的爆炸性增长, 传统方法在处理大规模代码样本时, 往往需要耗费大量的计算资源和时间, 海量的样本为恶意代码检测带来了巨大的挑战。在该背景下, 如何设计更好的检测方法成为网络安全领域亟需解决的问题之一。

恶意代码检测技术主要通过恶意代码的三种类型特征进行检测: 静态特征、动态特征以及两者组合而成的混合特征[5]。其中, 静态特征和动态特征的主要区别在于特征是否需要通过动态运行代码获得。早期恶意代码检测方法包括: 人工分析法、内容匹配法、统计分析方法等。人工分析法虽然检测精度高, 但需要大量的人力物力, 难以适合大规模分析的需要。内容匹配方法将恶意代码及其特征存入数据库中, 面对未知代码时, 通过将代码及其对应的特征在数据库中进行匹配, 进而对未知代码的类型进行判断。该方法虽然能实现自动化、批量化的检测, 但是随着代码样本的海量增长, 会导致数据库内存空间消耗变大, 匹配速度变慢, 且难以对未知的恶意代码进行检测。统计分析方法通过对代码信息进行挖掘和分析来实现恶意代码检测。相较于内容匹配方法, 该方法内存空间消耗小、检测速度快、能够在一定程度上对未知代码进行检测, 但对于部分代码样本和类别而言, 统计信息区分性能有限, 且难以对变体恶意代码样本进行检测。

机器学习方法在许多研究工作中被证实有着很好的分类效果[6], 因此研究者们也把机器学习方法引入恶意代码检测, 通过特征工程对恶意代码特征进行提取, 用于训练机器学习模型。作为机器学习方法的一个分支, 深度神经网络通过足够深层次的神经元去拟合数据的特征, 在性能上比其他机器学习方法有着更好的效果。因此, 基于深度学习的方法已逐步成为本领域研究的热点。

Chakkaravarthy 等人[7]对当时恶意代码检测的主流方法进行了综述, 但主要分析传统方法; Sihwail 等人[8]从静态分析方法、动态分析方法、混合方法、内

存分析方法对恶意代码检测方法进行综述; Ucci 等人[9]总结了许多能用于恶意代码检测的机器学习模型, 这些机器学习模型在静态分析和动态分析过程中都有具体的应用。近年来, 随着深度学习的快速发展, 基于深度学习的方法已成为恶意代码检测的主流方法之一。许多恶意代码检测综述[2,8,10]包含深度学习方法的介绍, 但介绍的内容相对有限。在现有的基于深度学习的综述中, 部分综述[11-12]局限于特定的环境; 部分综述[13-15]论述的深广度均有较大的限制, 没有反映近年来的研究进展。

目前, 随着深度学习在恶意代码检测领域的广泛应用, 方法类别也逐渐变得明晰。合理划分各个派别, 分析和探讨其背后的机理, 总结其优缺点, 能对未来工作起到激励和借鉴作用。基于此, 本文通过对基于深度学习的恶意代码检测方法进行综述, 旨在为研究人员了解恶意代码检测的基本原理和最新的恶意代码检测方法提供参考和帮助。

本文的主要贡献如下:

① 对基于深度学习的恶意代码检测方法进行了较为深入的讨论与对比分析, 弥补了已有综述在分析深度和广度上的不足, 也对近年来提出的方法进行了补充;

② 根据检测方法的应用场景、内部机理、网络结构、依赖特征, 本文将目前主要的方法分成了 5 个类别。这种分类有助于读者更好地理解和分析已有的工作;

③ 将计算机视觉和自然语言处理的深度神经网络运用到恶意代码检测领域, 恶意代码的表征会与常规图像、自然文本之间存在异同。本文对这些异同进行了详细的分析和讨论。

本文其余部分安排如下: 第 2 节主要介绍恶意代码检测技术使用的三种基本特征类型: 静态特征、动态特征和混合特征; 以及经历的三个重要发展阶段: 早期方法阶段、机器学习方法阶段、深度学习方法阶段; 介绍衡量模型性能的主要指标。第 3 到第 7 节分别根据恶意代码不同的特征角度, 对基于熵信息的法、基于图的方法、基于计算机视觉的方法、基于自然语言处理的方法、基于多维度特征融合的方法进行综述。第 8 节从恶意代码攻防对抗角度对梯度攻击方法、GAN(Generative Adversarial Networks)方法、强化学习方法进行介绍和分析。第 9 节讨论了当前恶意代码检测存在的问题和挑战, 以及未来可能的研究问题。第 10 节对全文进行总结。

2 恶意代码检测技术的发展

如图 1 所示, 恶意代码检测方法使用的特征主要包括: 静态特征、动态特征和由前两者组成的混合特征这三种基本类型; 方法的发展过程可以分为:

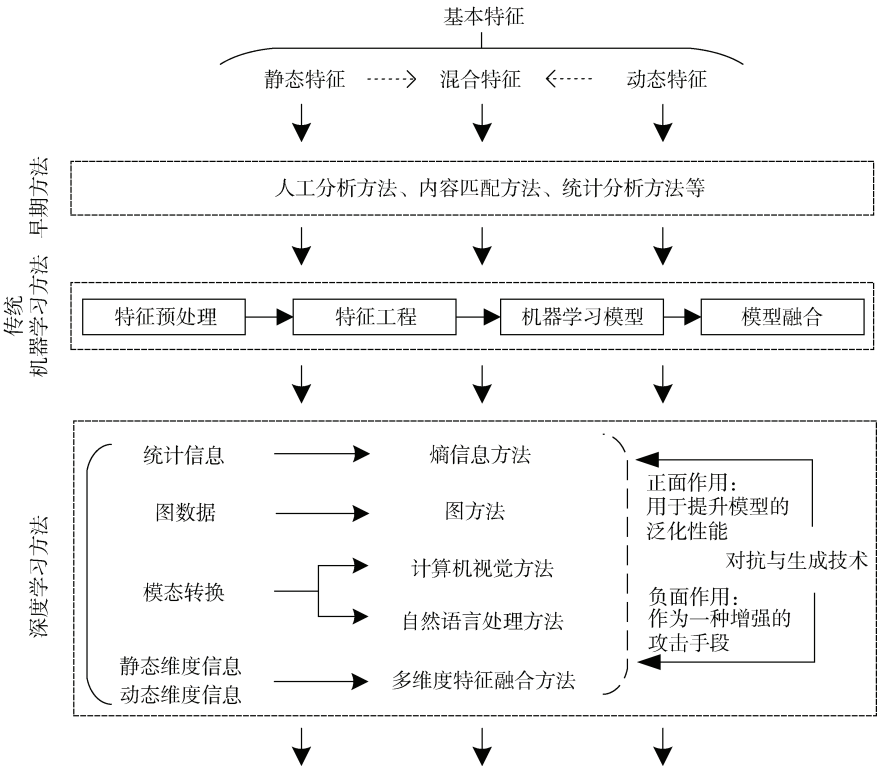

图 1 恶意代码检测方法各个发展阶段示意图

Figure 1 Various development stages Schematic diagram of malware detection methods

早期方法、传统机器学习方法和深度学习方法这三个发展阶段。分析恶意代码需要根据其特征进行分析, 而这三种基本特征类型几乎贯穿了恶意代码所有的发展阶段, 大部分方法所使用的特征都可以归类为其中一种基本特征类型。早期方法使用人工分析、摘要匹配、统计分析等方法进行识别; 传统机器学习通过数据挖掘、特征工程方法提取关键特征, 使用这些关键特征训练机器学习模型; 深度学习方法使用深度神经网络对特征进行学习。此外, 深度学习方法可以在学习原始样本的基础上, 通过生成与对抗技术生成更多的伪样本, 因为伪样本与真实样本的相似度很好, 因此可以使用伪样本对原始数据进行扩充和增强。虽然生成与对抗技术可以提升模型的泛化性能, 但设计者也能使用这项技术生成逃避率更高的恶意代码, 因此其负面作用值得警惕。

2.1 基本特征类型

在对恶意代码进行分析时, 需要从代码中抽取重要的特征进行分析。静态分析方法通过恶意代码的内容和结构等静态信息进行分析, 是恶意代码检测领域最经典的方法之一。静态分析方法的输入数据通常是从恶意代码源文件提取的特征, 这是可以直接获得的。尽管静态分析方法在许多恶意代码检测任务中都表现出很好的检测性能, 但该方法缺陷也很明显, 恶意代码设计者只需要对代码进行稍微改动就能提高恶意代码的逃避率, 使用添加无效字节、打包、加壳、混淆等方法让恶意代码更难被检测出来, 在很大程度上会影响恶意代码分析效果[16]。

使用环境平台运行代码, 通过代码在运行过程中产生、调用、更改信息的动态行为进行分析的方法, 称为动态分析方法[17]。恶意代码对系统有破坏性, 因此一般使用沙箱作为运行平台, 并通过技术手段收集代码在沙箱中运行的动作信息。这些动作信息通常具有时序性、前后文关联性, 因此可以通过动作信息来检测该代码是否为恶意代码。动态分析方法可以极大地减轻添加无效字节、加壳、混淆等方法对模型检测效果的影响, 但也存在缺陷: ①恶意代码运行时间开销大; ②设计者可以通过增加延迟来规避检测; ③恶意代码只有在合适的操作系统上才能进行动态分析, 如果操作系统不合适, 则无法运行代码和获得代码的动作信息。

在静态分析方法和动态分析方法的基础上, 研究者们提出混合分析方法[4,18], 即将恶意代码的静态特征和动态特征一同用于分析。混合分析方法从静态和动态两个维度进行结合分析, 分析效果往往会比使用单个维度分析效果要好。但是, 混合分析方法的特征获取和分析过程更加繁琐。在静态和动态特征不对称时, 性能往往会有所下降。

上述三种类型特征贯穿了恶意代码分析方法的全部发展阶段, 是分析方法重要的组成部分。目前, 几乎所有的分析方法都需要在至少一种类型特征上进行分析。尽管现在有很多能够自动学习特征的方法, 如何提升特征学习效果, 如何提升数据质量仍是值得思考的问题。高质量的数据和高效的学习策略能够提升检测方法的性能。

2.2 早期方法

早期的恶意代码分析方法有如下几类。

基于人工调试的方法。早期的检测方法主要以人工调试为主, 使用 IDA PRO 等反汇编工具, 对代码从内容和逻辑上进行分析与综合评定。这种方法虽然准确率高, 但是受限于人力、时间成本, 该方法只是适合小规模的恶意代码分析。

基于匹配的方法。通过大范围收集恶意代码来构建数据库, 将待检测代码放入数据库中进行匹配, 如果匹配成功, 则是恶意代码。该方法虽然在准确率上没有人工调试的方法高, 但是能在较短时间内对大批量的恶意代码进行检测。随着恶意代码数量的不断增加, 也需要更多的空间和时间用于数据库的构建与代码匹配, 研究者们开始使用 MD5 码对恶意代码进行编码, 通过 128 位的散列值对恶意代码进行标记。MD5 码有很高的标识性能, 将恶意代码转换成 MD5 码, 并用 MD5 码构建数据库, 极大地降低了构建数据库需要的容量以及提升匹配效率。

基于特征的方法。相关研究表明, 代码是否为恶意代码和其中的关键特征有较大的关联, 如 Opcode, PE head 等[19-21]。通过抽取这些特征并从统计学的角度进行分析, 能提升恶意代码的检测率。

2.3 传统机器学习方法

本文中, 传统机器学习方法指的是深度神经网络兴起之前的机器学习方法, 相关研究已经证实了一些传统机器学习方法有着较好的检测性能, 如支持向量机(Support Vector Machine, SVM)[22-23], K 近邻(K-Nearest Neighbor, KNN)[24-25]等。

Santos 等人[24]提出了结合动态分析和静态分析的恶意代码检测方法, 一方面使用 N-gram 模型提取操作码的二元特征, 之后计算特征的 TF-IDF 值作为序列的静态特征, 另一方面使用 API 调用序列, 通过特定的方法将序列转化成二进制数值作为序列的动态特征, 将这两种特征合并用于训练机器学习模型。Santos 等人[24]的实验证明, 尽管使用相对简单的机器学习模型(K 近邻、支持向量机、决策树、朴素贝

叶斯、贝叶斯网络)进行训练, 训练后的模型也有很好的检测效果。Zhong 等人[26]对恶意代码的静态和动态特征进行编码, 之后使用 K-means 方法对其进行聚类操作, 接着再对簇进行聚类操作生成子簇, 根据子簇之间的关联性构造簇的树结构。使用深层神经网络对树中的每个子簇进行训练, 从子簇训练结果中选择最好的作为该簇的结果, 并最终将所有簇的结果进行融合。在聚类和训练过程中使用并行计算的方式, 加快了训练和分析的速度。

传统机器学习方法具备检测准确率高、数据需求量少、解释性较强、训练和预测速度快等优点。但传统机器学习方法需要人工提取和构造特征, 因此数据处理环节工作量大, 并且特征工程的质量在很大程度上会影响该方法的检测性能。此外, 该方法的鲁棒性弱, 面对经过混淆操作的对抗样本时, 检测性能会大幅下降。在整个发展阶段中, 传统机器学习方法一方面是早期方法中的统计分析方法的延伸和扩充, 另一方面也为深度学习方法的兴起做了充分铺垫, 是承前启后的重要方法。

2.4 深度学习方法

深度学习方法在恶意代码检测中有着较为广泛的应用。一方面, 可以直接使用静态特征、动态特征、混合特征, 以及其他经过加工和处理过的特征训练深度模型。另一方面, 得益于深度学习方法在图像识别和文本处理上的成功, 研究者们尝试将代码特征转化成图像或类文本序列, 进而使用改进的图像分类和文本分类模型进行检测。此外, 通过生成和对抗方法可以进一步提升模型的鲁棒和泛化性能。深度学习方法较早期方法和传统机器学习方法而言, 有如下优点: ①能够自动提取特征: 深度网络有强大的特征提取能力, 因而对特征工程的依赖低; ②较强的泛化能力: 通过大量的神经元对特征进行拟合, 在遇到未知、变种的恶意代码样本时, 往往仍能保持较高的准确率; ③适合大数据处理: 可以使用 GPU 进行加速, 训练数据量越大, 检测效果往往越好; ④支持持续学习: 可以在训练好的模型上进行二次训练, 因此可以持续更新和改进模型, 以应对恶意代码不断变化。

2.5 模型性能主要衡量指标

目前衡量恶意代码检测模型的主要指标如图 2 所示。主流的检测任务包括两种类型: 其一, 判断其是否为恶意代码, 代码的标签仅有良性和恶性, 属于二分类任务; 其二, 判断恶意代码的家族或类别, 代码的标签为其所属的家族或类别的名称, 家族和类别往往超过 2 个, 属于多分类任务。对于二分类任务, 使用混淆矩阵衡量模型的性能; 对于多分类任务, 可将恶意代码所属的类别记为正类, 所有其他的类别记为负类, 通过此方法将多分类问题转化为若干个二分类问题, 并分别使用混淆矩阵衡量其效果。

在二分类问题中, 实际标签有真、假两种标签, 分别记为 *GroundTrue* 和 *GroundFalse*, 预测标签有真、假两种标签, 分别记为 *PredictTrue* 和 *PredictFalse*, 构造上述实际标签和预测标签的笛卡尔积, 得到四个元组:

(*GroundTrue*, *PredictTrue*)
(*GroundTrue*, *PredictFalse*)
(*GroundFalse*, *PredictTrue*)
(*GroundFalse*, *PredictFalse*)

分别对应真阳、真阴、假阳、假阴四种情况, 以这四种情况为基础, 计算衡量模型性能的主要指标, 包括准确率、错误率、精确率、召回率、F1 分数、真阳率、假阳率、受试者工作特征曲线、ROC 曲线下的面积等, 计算公式或方法如图 2 所示。

3 基于熵信息的方法

3.1 基本原理

将恶意代码转化成二进制形式, 以字节为单位对二进制代码进行切割并计算每个字节的数值。根据特定的长度, 将连续的字节按照长度进行分块, 根据每个数据块中所含字节数值的频率, 计算该块的熵值, 作为恶意代码的特征。熵值计算方法如公式 1 所示:

$$H(X) = -\sum_{i=0}^{n} p(x_i) \cdot \log_2 p(x_i) \tag{1}$$

其中, *H(X)*为数据块的熵值, x_i 表示块中数值为 i 的字节的频率, 字节数值的区间为[0,255], 因此 n 的数值为 255。

3.2 相关方法

Lyda 等人[27]最早提出使用熵信息对恶意代码进行检测。Bat-Erdene 等人[28]通过文件中特定块的熵信息, 来检测文件是否经过打包处理。Gibert 等人[29]使用文件熵信息在内的多种信息, 进行融合训练, 训练好的模型有很好的检测性能。Gibert 等人[30]使用小波变换对二进制恶意代码熵信息进行处理, 得到近似系数和差值系数, 接着使用卷积层+池化层的结构, 从这两种系数中提取信息。除了使用一维熵信息进行分析外, Vu 等人[31]使用恶意代码的熵值生成图像进行分析。他们将恶意代码块的熵值进行 MinMax 处理, 映射到[0,1]区间。接着使用 Sigmoid 型函数和二次函数对数据进行计算, 再乘以 255, 计算结果分

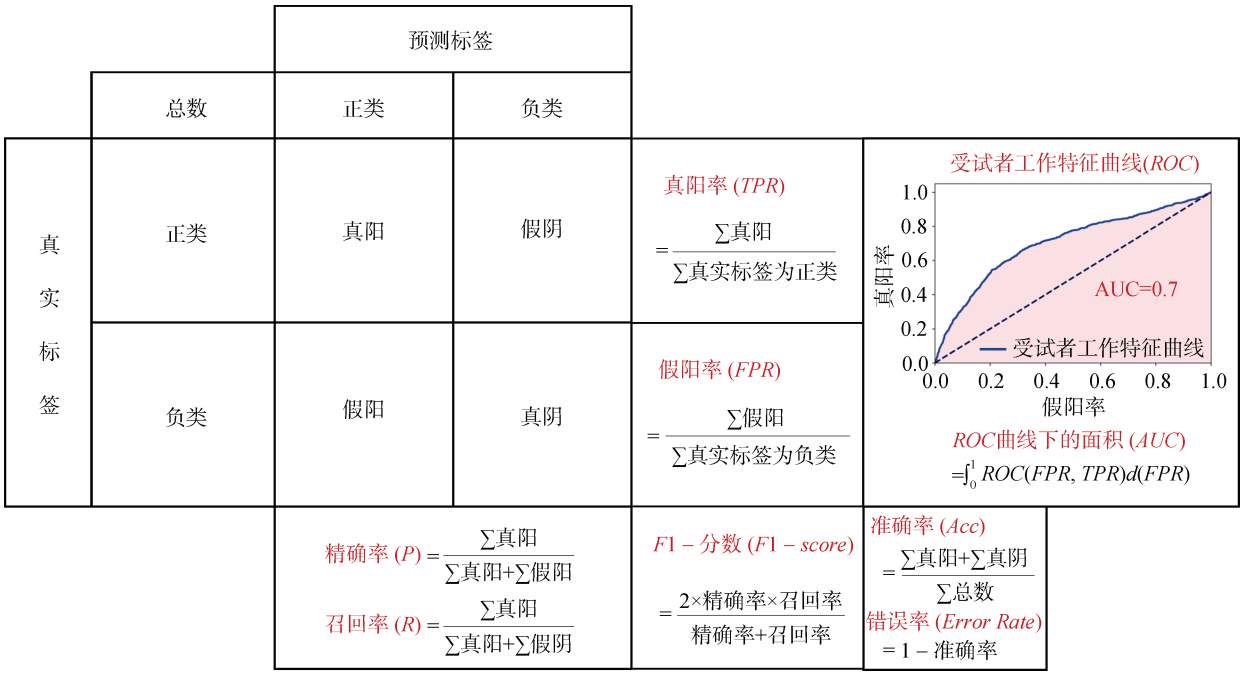

图 2　基于混淆矩阵的评估指标

Figure 2　Evaluation metrics based on confusion matrix

别作为 RGB 图像红色通道图像和蓝色通道图像。绿色通道图像使用句法信息构建。生成的图像分别使用 CNN 模型和 XGBoost 模型学习, CNN 模型能学到更好的效果。

3.3　讨论

从代码中提取熵信息的过程简单且高效，深度学习方法通过从熵信息中自动提取特征进行学习，进而实现恶意代码分类。该方法能够在很大程度上减轻无效代码插入、代码位置变换、加密与包装等混淆技术对检测器的影响。相较于支持向量机、决策树等机器学习方法而言，深度学习方法一方面借助 GPU 设备和批训练策略，降低训练过程的时间消耗和空间开销，因此能够对大量数据进行学习和分析，另一方面在得到新的训练样本时可以直接进行增量训练，而不需要同时使用新旧样本重新训练模型，以实现模型的更新迭代。但熵信息方法只关注代码字节级的统计特征，无法对代码的前后文语义进行捕捉，在一定程度上会影响检测效果。

目前熵信息方法主要应用在二进制代码以及恶意代码图像上。如果将熵信息方法应用在 API 调用序列等其他的恶意代码特征上，是否也会取得较好的检测效果？引入交叉熵、相对熵等方法，是否能够实现恶意代码样本之间的对比与匹配？这些都是值得进一步探讨的问题。

4　基于图的方法

4.1　基本原理

图是一种数据结构，往往以矩阵的形式呈现数据的结构和关系。图方法使用图数据结构对恶意代码特征进行表示，常用于恶意代码动态分析。图的节点表示实体(序列、文件等)；图的边表示各个实体之间的关联，用于体现代码调用实体的顺序。图方法相较于其他的特征工程方法而言，能更好地刻画代码的动态行为特征，并根据特征进行回溯分析。近几年来，随着图神经网络的兴起与发展，研究者们将特征转化成图数据，使用图模型进行分析。

4.2　相关方法

Li 等人[32]从 API 序列的名字和参数中抽取语义信息，根据 API 调用的顺序构建 API 序列调用图，并使用图卷积网络(Graph Convolutional Networks, GCN)[33]和图注意力网络(Graph Attention Network, GAT) [34]等方法学习 API 调用图的结构信息。Pei 等人[35]从安卓 API 序列中抽取词级和字符级特征，根据这些特征构建图并生成图表征，之后用图卷积网络提取表征特征，再使用 IndRNN [36]学习前后文特征的语义关联信息。Zhang 等人[37]使用 API 分析文档作为原始数据，从里面抽取实体和关系，根据实体和关系构建 API 序列图—APIGraph，通过序列嵌入和聚类方法对 API 序列图进行增强，增强后的图

能够对 API 序列更好地表征和刻画 API 序列之间的联系。该方法是一种数据增强的方法, 在增强的数据上进行训练, 能够提升其他恶意代码检测器的检测性能。

除了序列调用图, 溯源图方法也得到研究者的关注。溯源图(Provenance Graph)是一种用于表示实体(例如文件、进程和网络连接)及其之间关系的有向图。基于溯源图的恶意代码检测方法是一种利用恶意代码的行为特征和关联关系进行检测的技术。在恶意代码检测领域, 溯源图可以帮助我们理解恶意代码的行为模式, 从而识别和阻止潜在的恶意活动。Wang 等人[38]使用进程、文件、套接字作为图的顶点, 使用各个实体间的数据依赖和控制依赖作为边。根据溯源数据构建图, 根据频率数据选择 K 条最为罕见的运行流程, 从图中抽取这 K 个运行流程所包含的顶点和边, 之后使用嵌入的方法对图数据进行表征, 最后使用局部异常因子算法(Local Outlier Factor, LOF)对溯源图进行检测。Wang 等人[39]设计一种从节点层面对溯源图进行无监督学习的检测模型。该模型以 GraphSAGE 为基本框架, 对邻居节点进行采样和聚合, 最终通过富集信息的节点来预测类别。此外, 该模型的学习模式为归纳学习(Inductive Learning), 相较于 GCN 的转导学习模式(Transductive Learning)而言, 它不需要在训练过程中使用到测试数据, 在训练完后, 可以对未见过的测试数据进行预测。

4.3 讨论

图方法能够有效地捕捉恶意代码之间的关系和交互模式, 包括函数调用关系、数据依赖关系、控制流程等, 这使得图方法能够更全面地表示恶意代码的行为特征。当恶意代码被混淆时, 其图结构并不会发生太大的改动, 因此对检测器的影响小。相较于向量嵌入方法而言, 图方法能够对关键字的含义进行更好地体现, 比如通过不同的有向边表示选择指令, 通过有环图表示循环指令。因此, 图方法也是未来研究的主流方法之一, 如何选择更合适的实体, 以及在实体之间建立更好地联系, 是未来图方法值得思考的问题。

但是, 将恶意代码表示为图结构需要对代码进行解析、关键数据提取、数据结构重构等环节, 数据处理工作比较复杂。此外, 构造的图往往包含大量的节点和边, 这使得图的维度较高, 需要更多的存储资源、计算资源和时间成本进行分析。

5 基于计算机视觉的方法

Nataraj 等人[40]最早引入计算机视觉的方法解决恶意代码分类问题, 他们将文件的二进制代码, 按照每 8 个比特生成一个向量, 然后将向量转换为灰度图, 通过提取图像特征对图像进行分类。基于计算机视觉的方法为后续基于深度学习的静态图像分析方法提供了理论基础, 其主要架构如图 3 所示。

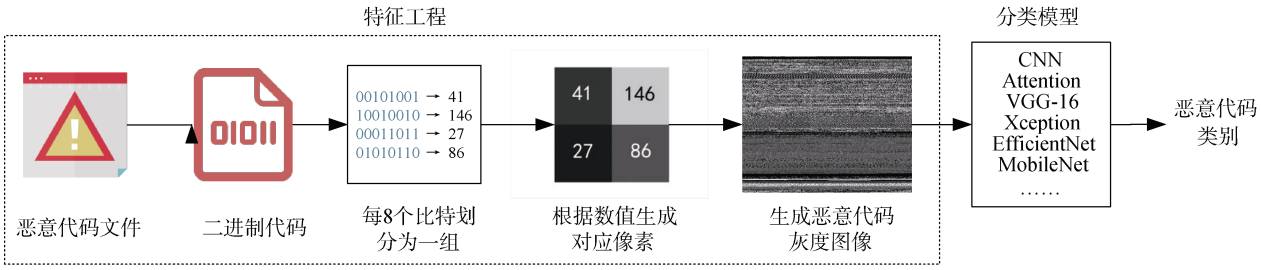

图 3 基于计算机视觉方法的主流分析框架

Figure 3 Mainstream analysis framework based on computer vision methods

5.1 基本原理

通过抽取恶意代码中的信息来生成图片。一种常用的方法, 是将恶意代码转化成二进制的形式, 将每 8 位比特分为 1 组, 每组可以用于表示 1 个像素, 8 比特可以表示 256 种信息, 正好对应灰度图像 256 个灰度等级。通过这个方法可以将恶意代码转化成图像, 作为其视觉表示。使用这些图像训练分类模型, 训练好的模型可以对图像进行分类, 进而对恶意代码进行分类。除此之外, 部分研究者设计了其他的恶意代码图像表示方法, 比如使用部分关键信息 Hash 码进行恶意代码图像生成[41]。

5.2 相关方法

随着深度神经网络在图像分类任务上越来越成熟, 在 Nataraj 等人[40]工作的基础上, 许多面向恶意代码灰度图像的模型和方法相继提出。其中最为经典的就是使用卷积神经网络(Convolutional Neural Networks, CNN)架构对图像进行分析。Makandar 等人[42]使用 Gabor 小波和 GIST 方法提取图像的纹理特征, 之后使用卷积神经网络对纹理特征进行分类。Zhao 等人[43]从图像中切割出信息量充足的图像区域,

并进行纹理特征提取, 之后使用多层卷积网络对特征进行训练和学习。Gibert 等人[44]提出了一种与文件无关的深度学习方法, 将恶意代码转换成灰色图像, 并使用多个池化+卷积的组合进行特征提取, 最终通过全连接层连接各个代码类别。此外, Kalash 等人[45]、Mourtaji 等人[46]尝试用深度神经网络 CNN 应用在灰度图的分类任务上, 进而实现恶意代码的分类。

基于 CNN 架构的模型已经取得了不错的检测效果, 在此基础上融合多个图像分类模型能进一步提升了模型的检测性能。Vasan 等人[47]使用经过 ImageNet[48]预训练的 VGG16 模型[49]和 Resnet50 模型[50], 通过模型集成的方法对恶意代码图像进行分类。Parihar 等人[51]使用集成学习的方法选择图像分类领域性能较强的 Resnet50[50], Xception[52] 和 EfficientNet-B4[53]模型, 在模型最后使用多层全连接层对 3 个模型输出信息进行融合。结果证明, 经过融合后的模型相较单一模型而言, 有更好的检测效果。

除了使用二进制代码生成恶意代码图像外, 研究者们也在不断尝试使用其他的恶意代码静态特征信息, 来进行图像的生成和检测。Ni 等人[41]使用恶意代码中 Opcode 序列生成图像, 他们构造一种名为 SimHash 的哈希映射, 从恶意代码中抽取 Opcode 序列, 经过 SimHash 映射后得到哈希向量, 进而使用哈希向量生成对应图片, 通过多个卷积层和池化层抽取图片特征。Nguyen 等人[54]通过 BE－PUM 工具获取二进制代码的数据流图, 计算数据流图的邻接矩阵, 并由矩阵生成图片, 最后使用 CNN 架构对图片进行分类。Yuan 等人[55]使用马尔可夫转移矩阵生成恶意代码图像。将代码转化为字节形式, 统计代码中当前字节和下一个字节之间的转移数量, 并计算转移概率, 每个字节共有 256 种表示形式, 则字节与字节之间的转移概率矩阵大小为 256×256, 最终将矩阵转化成图片的形式, 使用 CNN 网络架构对其进行分类。

除了将静态分析信息转化成图像外, 研究者们也尝试将恶意代码的动态行为信息转化成图像并进行检测。在动态分析方法上, Tang 等人[56]将 API 调用序列转化成图像进行分析。他们将 API 调用序列分为 n 个主要的类别, 运行时间被平分成 n 个区间, 统计时间区间中各个 API 类别的数量, 根据数量生成相应的像素, 最终得到一张 $n\times n$ 的恶意序列图像, 然后使用 CNN 网络对图像进行分类。Tobiyama 等人[57]使用 RNN 网络架构提取 API 序列中的语义特征, 在提取完特征后, 没有使用多层全连接层连接分类器类别, 而是将提取的特征生成图像, 再使用 CNN 分类器对图像进行分类。

Transformer[58]最早用于机器翻译领域, 通过多头注意力机制来捕获更加丰富的语义信息, 在 Transformer 的基础上, BERT[59], GPT[60], ViT[61]等更加强大的模型也相继被提出, 运用的领域也越来越广泛。Yakura 等人[62]在深度 CNN 网络之后加入了注意力机制层, 能对图片中重点的区域给予更多的关注, 以提高模型的检测效果。Seneviratne 等人[63]将 Transform 架构运用到基于视觉的方法上, 使用自监督的 ViT 模型对安卓恶意代码图像进行分类, 与其他主流的图像分类模型相比有更好的效果。代表性工作总结于表 1。

表 1　部分基于计算机视觉的代表性工作性能对比

Table 1　Performance comparison of partial representative jobs based on computer vision

论文	发表年份	方法	在 BIG2015[64]上的检测率	在 Malimg[40]上的检测准确率
Nataraj et al. [40]	2011	GIST	—	98%
Simonyan et al. [49]*	2014	VGG-16	97.79%	98.18%
Simonyan et al. [49]*	2014	VGG-19	97.24%	95.72%
He et al. [50] *	2016	ResNet50	99.08%	98.93%
Kalash et al. [45]	2018	CNN	99.97%	98.52%
Tan et al. [53] *	2019	EfficientNet-B4	99.17%	98.93%
Howard et al. [65]*	2019	MobileNetV3	99.08%	99.36%
Mourtaji et al. [46]	2019	CNN	97.02%	—
Gibert et al. [44]	2019	CNN	97.49%	98.48%
Vasan et al. [47]	2020	VGG-16	—	98.82%
Parihar et al. [51]	2022	ResNet50, Xception, and EfficientNet-B4	—	99.43%

注: 标“*”的为基线模型

5.3 讨论

基于计算机视觉的方法是使用特定的方式将恶意代码以及其特征变成图像, 通过图像分类模型进行分类, 进而对恶意代码进行分类。目前, 基于深度学习方法的图像分类任务已经发展得较为成熟, 这种方法在实验中有非常好的效果, 检测准确率基本都在 95%以上。但是该方法也存在着一些缺点, 由于大部分基于计算机视觉的方法是静态分析方法, 它也继承了静态分析法的缺点, 遇到加壳、混淆、冗余填充的恶意代码时, 该种方法的检测性能有所下降。除此以外, 模型的解释性差。为什么将恶意代码变成图片能提高检测效果? 其内在的原理是什么? 尚未有文献对以上问题进行较好的解释。

恶意代码图像与自然图像的差异性很大。自然图像色彩的聚集性与连续性很强, 以图 4 为例, 图像颜色布局自上而下分别为: 天空的浅蓝色, 海水的深蓝色、绿色、白色, 沙滩的黄色。对于每一个区域而言, 中间的颜色较深, 边缘颜色较浅。区域中的颜色会有些许偏差, 但是颜色相对集中, 且颜色的变化呈现连续性。对于恶意代码图像而言, 从人的视觉角度来观察, 图像几乎无法提供任何有用的信息。此外每个像素点的颜色是相对独立的, 与周围的颜色关联不大, 也无法体现颜色变化的趋势, 因此恶意代码图像中的每个像素体现出很强的离散性。从图像的统计特征层面进行分析, 将自然图像和恶意代码图像灰度化, 自然图像的方差为 2482, 恶意代码图像的方差为 5660, 可见恶意代码图像的波动程度更大。在图像频谱方面, 自然图像无论实在水平方向还是在垂直方向, 其频谱变化都相对平和; 恶意代码图像在水平方向的频谱曲线近似白噪声, 其垂直方向的频谱波动非常大且剧烈。

其次, 将自然图像黑白化, 或附加微弱干扰(如: 加噪声、调整色调、扭曲等), 对自然图像信息表达的影响都不大, 不影响人对图像关键内容的识别。对于恶意代码图像而言, 图像中的每个像素点, 都是二进制代码的单向表示。即使是对一个像素点进行修改, 都会对代码的内容进行修改, 如果被修改的内容是注释信息, 那么对代码没有太大的影响, 但如果被修改的是关键信息则会导致代码无法正常运行。尽管加噪声会极大地改变恶意代码的语义信息, 但目前也有相关研究表明(见第 8 节): 使用正常的恶意代码二进制图像和经过生成与对抗方法得到的恶意代码二进制对抗图像一同用于检测模型的训练, 可以提升检测模型的泛化性能。

在未来工作中, 除了研究恶意代码图像的解释性外, 如何在保持高准确率的前提下, 增强模型的泛化能力, 提高模型对对抗样本的检测率, 也是基于计算机视觉的检测方法的重要课题。

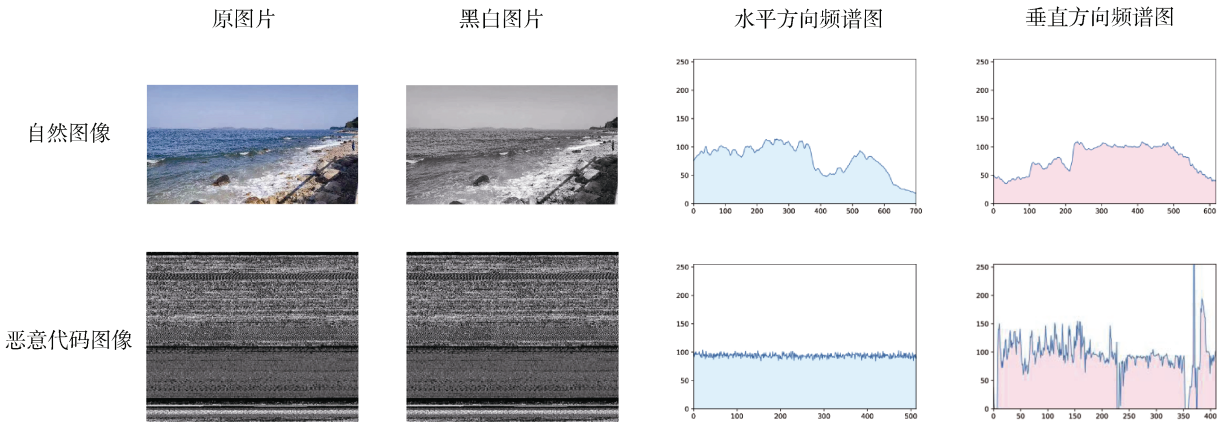

图 4 自然图像与恶意代码图像的差异

Figure 4 The difference between natural images and malware images

6 基于自然语言处理的方法

直观上理解, 自然语言处理方法和恶意代码分析代码相差甚远, 之所以能将 NLP 技术运用到恶意代码分析上, 主要缘于恶意代码分析过程中的特性: 一方面, 在动态分析的过程中收集到的动作信息有很强的序列性。经过编码后的自然语言本质上也是序列, 因此在序列分类以及其他的下游任务上, NLP 领域很多模型都表现出很好的效果。因此, 可以通过迁移和改进 NLP 模型, 以设计用于恶意代码检测的模型。另一方面, 恶意代码本身也是一种结构严谨、内容完整、语法正确的“计算机语言”, 与自然语言也有共通的地方, 可以通过对“计算机语言”抽取关键信息、编码、表征, 进而使用深度神经网络学习表

征信息, 从而实现代码分类任务。在 6.2 节和 6.3 节中, 我们会对恶意代码表征方法和 API 序列分析方法分别进行工作介绍和讨论。

图 5 展示了一种主流的面向 API 调用序列的恶意代码检测算法。首先进行数据收集和获取操作, 接着使用恶意代码分析平台(如: Cuckoo 沙箱[66]、CAPE 沙箱[67])运行恶意代码并收集其动态行为信息, 这些信息一般以 json 格式的文件进行记录, 从文件中抽取 API 调用信息, 使用 TF-IDF、n-gram、词嵌入等方法提取特征, 再使用深度神经网络学习这些特征。

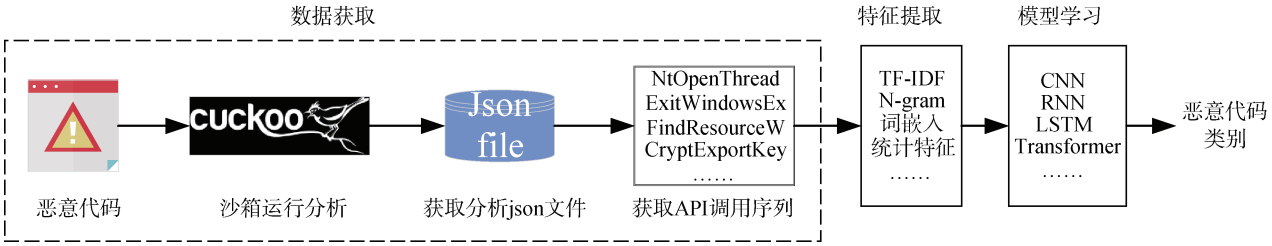

图 5 基于自然语言处理方法的主流分析框架

Figure 5 Mainstream Analysis framework based on natural language processing methods

6.1 基本原理

N-gram[68]。序列之间存在时间先后联系。在特征工程中, 可以将相邻的序列打包并编码, 作为新的训练数据。*n*-grams 中的 *n* 表示的是组的大小, 例如 *n* 为 4 时, 就把前后相连的 4 个序列打包成一组。TextCNN[69-70]的设计也是参考了这个思想, 选用不同的 *n* 提取词元, 并对信息进行融合, 在自然语言处理以及恶意代码的检测上都有较好的效果。

TF-IDF[69]。以文本处理任务为例, *TF-IDF* 由两部分组成, *TF* 表示词频, 即当前词语在文章中的频率。如果一个词的出现频率越高, 那么有比较大的概率是重要的词; *IDF* 表示逆文档频率, 有些词出现的频率很高, 但是对于文本任务而言效果不大, 因此定义 *IDF* 作为词语普通性的度量, 普通性越强, 重要程度越低。

$$TF_i = \frac{API_i}{\sum_{i=0}^{n} API_i} \tag{2}$$

$$IDF_i = \log\left(\frac{|Call|}{|\{API_i \in Call\}|+1}\right) \tag{3}$$

$$TF-IDF_i = TF_i \times IDF_i \tag{4}$$

在恶意代码分类任务中, 以 API 调用序列为例, 我们通过计算 API 序列频率和逆频率来计算每条 API 序列的 *TF-IDF* 值。API 序列频率计算如公式(2)所示, 假设 API 序列共有 *n* 种, 分别用 1 到 *n* 进行标识, *i* 表示其中一种类型的 API 序列, *i* 的序列频率等于 *i* 的序列的数量除以所用序列的数量; API 的逆文档频率计算如公式(3)所示, 每次调用过程包含多个 API 序列, $|Call|$表示调用总次数, $|\{API_i \in Call\}|$表示含有第 *i* 种类型的 API 序列的调用次数, 后面加 1 是为了防止分母为 0 的情况。第 *i* 种 API 序列的 *TF-IDF* 值等于其对应的 *TF* 和 *IDF* 值相乘, 如公式(4)所示。

Embedding[71]。自然语言处理任务中, 如果对每一个词都进行独热编码操作, 不仅需要非常大的内存开销, 而且各个字之间的相似程度也无法衡量。词嵌入方法是将高维的数据投影成低维数据。用一个向量来表示字, 解决了占用空间大, 矩阵稀疏等问题。对于动态分析而言, 我们先对序列进行编号, 之后使用一个低维向量对这个编号进行表示, 向量的值会在训练中逐步调整。

6.2 相关方法

6.2.1 恶意代码表征方法

Cakir 等人[72]对恶意代码中的 opcode 生成词向量, 并用深度神经网络提取其中特征, 最后使用梯度提升方法提升分类模型的性能。Raff 等人[73]以字节为单位对代码进行划分, 使用嵌入层为每段字节生成表征, 接着使用卷积层和最大池化层从提取表征信息, 最后通过全连接层连接输出。Krčál 等人[74]在他们的基础上进行改进, 在提取表征信息部分, 依次使用 2 个步长为 4 的 32 通道卷积层, 步长为 4 的最大池化层, 2 个步长为 8 的 16 通道卷积层, 全局平均池化层进行信息提取。提取后的特征依次输入含有 192、168 和 128 个单元的全连接层, 最终连接模型输出。增加网络层数和训练参数能在一定程度上提升模型的检测性能。但是, 部分助记符难以用词嵌入方法进行表征, 比如循环和判断, 循环指令的主要作用是引导程序对部分代码进行重复执行, 用词嵌入向量方法无法表示这一过程; 判断指令的主要作用是引导程序选择一个或多个分支执行, 对于不执行的部分仍要进行编码并进行分析, 这也是不合适的。

嵌入表征只是对特征的一个表征, 但对于恶意代码而言, 仅使用嵌入表征无法对前后文特征进行挖掘。因此, 研究者们尝试设计包含前后文语义信息的嵌入表征。Zhang 等人[75]先使用 N-gram 方法生成词元, 再对每个词元生成嵌入表征, 后面相继使用含有自注意力机制(self-attention)的 CNN 和 BiLSTM 提取信息。Sung 等人[76]提出了使用 FastText[77-78]的方法训练每个 opcode 和 API 函数的词嵌入向量, FastText 对特征进行字符级的 N-gram 处理, 并为其生成表征。该方法能更好地学习 opcode 和 API 函数的特征与前后语义关联, 且词表小, 增加的计算时间和空间消耗少, 在恶意代码检测效果上优于词嵌入和独热编码(one-hot)的表示方法。

6.2.2 API 序列分析方法

Pascanu 等人[79]首次提出使用自然语言处理方法进行恶意代码检测。在对 API 序列进行预处理和编码后, 采用了 ESN[80]和 RNN 模型提取内在特征。然后, 使用逻辑回归和多层感知器作为分类器, 最终输出为恶意代码的预测概率。与传统神经网络类似, ESN 网络主要由输入层、隐藏层和输出层组成, 但在训练过程中隐藏层的参数不能调整, 唯一可调的参数是输出权重, 因此收敛速度慢且学习效果不佳。对于长序列分析, RNN 模型容易出现梯度消失和梯度爆炸现象。

后来, 以 RNN 为基本框架的 GRU 和 LSTM 模型相继被提出, 在一定程度上缓解了梯度消失和梯度爆炸现象。Maniath 等人[81]和 Catak 等人[82]分别使用 GRU 和 LSTM 作为主干设计检测模型。实验证明, GRU 和 LSTM 设计模型的检测性能优于传统的 RNN 模型。Athiwaratkun 等人[83]分别使用 LSTM[84]、GRU[85]和 CNN 模块替换基于 Pascanu 工作的原始 ESN 和 RNN 模型。通过对比三者的性能, 发现使用 LSTM 模型可以获得更好的结果。GRU 和 LSTM 模型解决了部分 RNN 模型的缺陷, 但也存在诸如长序列分析能力差、无法并行运行等缺点。

除了 RNN 类模型, CNN 类模型和 CNN+RNN 类模型也表现出较好的检测性能。Qin 等人[70]使用 TextCNN 模型检测恶意代码动态 API 序列, 先为各个 API 序列进行编码, 之后使用嵌入层为其生成嵌入向量, 再使用多个卷积核大小不同, 通道数相同的 1 维卷积层提取嵌入层的信息, 最后使用最大池化层对卷积结果进行投影和拼接。该模型结构简单, 计算量较小, 但也有良好的检测性能。Kolosnjaji 等人[86]提出了一个结合 CNN 和 LSTM 的模型。首先, 用 3-gram 模型预处理 API 序列, 将三个连续的 API 序列组合成一个整体进行编码和数据对齐。卷积层中两个连续的 3×60 大小的卷积核和最大池化层提取信息, 接着使用 LSTM 层捕获上下文语义信息。最后, 通过全连接层和 softmax 方法连接输出类别。

Li 等人[87]提出了一个 CNN-BiLSTM 网络来检测恶意软件。首先, 从 API 序列中提取 API 嵌入特征和语义链。对于 API 嵌入特征, 使用大小为 3、4 和 5 的核进行 1 维卷积, 然后将结果拼接; 对于语义链, 首先进行词嵌入, 然后使用一维卷积层提取特征。将这两种特征提取结果进行拼接并输入进 BiLSTM 层, 接下来依次连接到两个连续的全连接层, 最后连接到每个输出类别。

Transformer[58]通过注意力机制捕获语义信息。通过利用大规模数据预训练多层 Transformer, 预训练模型通过简单的微调操作在其他下游任务上表现出很好的性能。在恶意软件动态分析领域, 分析人员试图使用大量的 API 序列来预训练模型, 以便训练后的模型具有更好的检测效果。Demirkiran 等人[88]使用基于 Transformer 结构的 CANINE 和 BERT 模型, 分别在文本语料库和 API 序列上进行预训练, 然后将预训练模型应用于 API 序列检测。Xu 等人[89]在 BERT 模型[59]的基础上提出了一种新颖的预训练模型 MalBERT。首先, 他们按照 NLP 预处理方法对 API 序列进行清洗、裁剪和对齐。其次, 使用标记嵌入(Token Embedding)、段落嵌入(Segment Embedding)以及位置嵌入(Position Embedding)来表示 API 序列。他们使用 12 个依次相连的 Transformer 从嵌入向量中提取特征, 并将输出结果进行拼接。最后, 他们使用 API 序列通过掩码语言模型(Masked Language Model, MLM)和下一句预测(Next Sentence Prediction, NSP)方法对模型进行预训练[59]。与基于 RNN 的方法相比, Transformer 可以进行并行计算和分析更长的序列, 但运行时间随序列长度呈多项式增长。BERT 模型的输入长度限制为 512, 但许多动态分析序列的长度超过此限制, 有限的表征将影响模型的检测性能。

动态序列分类任务研究已经比较成熟, 但是无法对异常序列所在的具体区间范围进行检测。Huang 等人[90]将注意力机制应用到了分类任务的下游任务——序列标注任务上, 尝试对恶意的 API 序列进行标注。为此, 他们设计一种序列到序列的网络(Seq2Seq), 对代码调用 API 的情况进行分析, 并使用特定的标签, 以标注代码在运行过程中产生的恶意行为。他们也是最先将序列标注方法运用到恶意代码分析的团队。

考虑到该类别中的各模型使用的数据集差异大, 难以对各模型性能进行直观比较。为此, 我们对该类别的主要模型进行复现, 并使用 Aliyun 数据集[91]在恶意代码多分类任务和二分类任务(良性代码和恶性代码)上训练和测试复现模型。除特别说明外, 本文对关键参数进行统一, 词嵌入维度为 128, 学习率为 0.001, 批大小为 64, 输入内容为 API 调用序列且截断长度为 1000。模型复现结果总结于表 2。

表 2 部分基于自然语言处理的代表性工作性能对比

Table 2 Performance comparison of partial representative jobs based on natural language processing

论文	发表年份	方法	在 Aliyun 数据集[91]上的多分类任务性能	在 Aliyun 数据集[91]上的二分类任务性能
Bengio et al. [92]*	1993	RNN	74.66%	89.70%
Hochreiter et al. [84]*	1997	LSTM	85.17%	95.32%
Schuster et al. [93]*	2005	BiLSTM	83.30%	95.10%
Chung et al. [85]*	2014	BiGRU	83.15%	94.96%
Chung et al. [85]*	2014	GRU	84.52%	95.10%
Kolosnjaji et al. [86]	2016	RNN+CNN	78.74%	93.02%
Huang et al. [94]	2016	MLP	85.03%	95.75%
Athiwaratkun et al. [83]	2017	LSTM/GRU	80.63%	94.02%
Qin et al. [70]	2020	TextCNN	87.54%	97.12%
Catak et al. [82]	2020	TF-IDF, LSTM	84.59%	95.46%
Xu et al. [89]	2021	BERT	80.20%	81.22%
Li et al. [87]	2022	CNN+BiLSTM	83.95%	95.68%

注: 标“*”的为基线模型。考虑到 Xu et al. [89]模型较大, 在复现时 batch_size 设置为 24。

6.3 讨论

6.3.1 恶意代码表征方法

恶意代码特征表征方法是一种简单且有效的方法, 相较于其他方法而言, 并不需要复杂的数据处理操作。为代码生成嵌入向量后, 使用深度学习网络自动学习其中的特征, 因此该方法是一个端到端的方法。但是, 该表征方法也有局限性。相较于自然文本而言, 恶意代码的长度往往更长, 且不同恶意代码的长度差异大, 这也直接导致恶意代码的定长阈值并不容易设置。大的阈值可以提取更多有用的信息, 降低信息丢失的比例, 但也会导致冗余填充的信息变多, 加大运行时间; 小的阈值信息丢失的比例大, 但是填充的冗余信息少, 运行时间短。部分代码难以用词嵌入方法进行表征, 比如循环和判断, 循环指令的主要作用是引导程序对部分代码进行重复执行, 用词嵌入向量方法无法表示这一过程; 判断指令的主要作用是引导程序选择一个或多个分支执行, 对于不执行的部分仍要进行编码并进行分析, 这也是不合适的。

6.3.2 API 序列分析方法

相较于静态分析, 基于 API 序列的分析方法能够极大降低加壳、混淆和添加冗余内容等方法对检测效果产生的影响。但该方法同样存在一些缺点: API 序列检测需要不小的内存和时间开销; 部分恶意代码要在特定的平台上才能运行; 设计者可以通过设置延时以避开检测; RNN 类模型不能并行运行, 训练时间开销大等。

为了能更好地分析自然文本和恶意代码动态序列之间的异同, 本文对自然语言数据集(THUCNews)[95]和 API 序列数据集(阿里云安全恶意程序检测数据集)[91]进行统计分析, 结果如表 3 所示。

与自然文本相比, API 序列长度更长, 但所需的词表更小。词表小, 则不需要过于复杂的网络架构, 都能很好地提取其中的特征。序列长, 在使用 RNN 类和 Attention 类架构的模型分析时, 训练时间会随序列长度的增加而显著提升。此外, 词表中的每个 API 序列都有对应的含义, 因此也不需要进行停用词去除操作。

表 3 自然语言和 API 序列的统计信息比较

Table 3 Statistical comparison of natural language and API sequences

	自然语言	动态分析序列
词表大小	4762	304
长度范围	3-38	$1\sim8.9\times10^6$
长度均值	19.21	6467
Q1 分位数	17	238
中位数	19	1770
Q3 分位数	22	7280
方差	14.93	3.7×10^8

目前, Transformer 是 NLP 领域的热点模型之一, 但在 API 序列动态分析上, 性能并不理想。除了检测性能不如传统的 CNN 类、RNN 类模型外, Transformer 的运行时间随序列长度呈现多项式增长。API 序列的词表非常小, 序列并不需要进行分词处理, 也不需要用如此复杂的模型对 API 序列进行表征。因此基于 Transformer 的模型是否适合 API 序列动态分析任务, 仍需要进一步研究。尽管如此, 以 Transformer 为基础的 BERT 模型, 给我们带来了新的思考方向。BERT 借助海量的数据, 通过无监督学习的方式, 学习到每个词的表征, 即预训练模型方法。处理下游任务时, 在经过预训练的表征上进行微调, 能加快训练收敛速度和提升学习效果。如何设计面向 API 序列的预训练模型也是未来的研究热点之一, 其中最为关键的是无监督学习任务和模型架构的设计。

7 基于多维度特征融合的方法

目前, 已有不少学者通过对多种特征进行融合来进一步提升模型的性能, 但这些特征全来自于静态分析或动态分析的其中一种。对于基本类型相同的特征, 其所表达的信息相似度较高, 对模型性能的提升有限。

Radford 等人[96]提出了 CLIP(Contrastive Language-Image Pre-Training)模型, 是多模态领域早期经典的模型之一, 通过从网上获取的海量的图像和句子训练模型。实验证明, CLIP 模型融合了图像模态和文本模态的信息, 在多个数据集上表现出优异的图像分类性能。受此启发, 研究人员们尝试使用静态特征和动态特征进行分析, 因为这两种特征是从两种不同的视角和维度对恶意代码的反映, 因此本文中将这种方法称之为多维度特征融合。

7.1 基本原理

多维度特征融合的方法通过结合静态维度特征(如: Opcode 码、二进制灰度图像等)和动态维度特征(如: API 调用序列、进程行为等)进行分析。多维度特征融合方法虽然在特征处理上较前面几类方法而言, 并没有呈现太大的不同, 但是结合两种不同维度的特征进行分析可以为恶意代码的检测提供更加全面的信息。

7.2 相关方法

静态维度特征与动态维度特征在获取途径、格式、内容上均呈现较大的差异, 但是大部分特征都是以离散数据的形式表示, 因此对特征进行编码, 并用向量进行表征, 成为多维度特征融合方法中特征表示的主流方式。Xu 等人[97]先对静态特征和部分动态特征编码成向量形式, 使用深度自编码器对特征进行学习。每个自编码器包含多个依次相连的受限玻尔兹曼机(Restricted Boltzmann Machine, RBM), 用以学习特征的表征。另外一部分动态特征为 API 调用序列, 将进程视为节点, API 调用序列视为各个节点之间的关联, 以此构建图数据, 作为 API 序列的表征。得到各个特征的表征后, 需要使用分类方法根据表征进行分类。文章通过使用不同的核, 计算各个特征表示之间的相似度, 并根据相似度构造核矩阵, 最终将各个特征的核矩阵进行拼接, 并使用 SVM 进行分类。Gibert 等人[98]使用嵌入方法对动态特征和静态特征进行表征。他们使用三种类型的特征作为模型的输入: API 调用信息、字节信息、opcode 码信息, 对这些信息分别进行嵌入操作, 再通过卷积层+池化层的结构提取特征。最终将这三种输入特征提取的信息进行融合, 通过全连接层连接各个输出类别。表征方法简单且实用, 如果能提升数据表征质量, 模型的性能也将进一步提升。

得益于深度学习技术在图像分类和文本分类上取得了巨大的成功, 研究者们尝试将特征转化成图像或类文本序列, 借助图像分类和文本分类模型对特征进行分析, 进而提高模型的准确率。Huang 等人[99]提出了将恶意代码的静态和动信息转化成图像进行检测的方法。图像由两部分组成, 上半部分: 使用恶意代码的静态信息生成的图像; 下半部分: 将恶意代码的 API 调用序列按照一定的规则映射成特定的像素, 并生成图片。Zhang 等人[100]从恶意代码中提取 opcode 码作为静态特征, 通过计算每两个 opcode 码之间的关联性得到 opcode 码的词元矩阵, 考虑到维度过高带来计算量过大的问题, 文中使用主成分分析方法(Principal Component Analysis, PCA)方法进行降维处理, 使用降维后的词元矩阵生成图像。对图像进行卷积和池化操作, 再连接含有 Softmax 的全连接层, 其结果作为静态分析部分的输出。另一方面使用 API 序列作为动态分析特征, 通过计算 API 序列中各个调用的频率信息, 并将信息输入进神经网络, 其结果作为动态分析部分的输出。最后将静态分析信息和动态分析信息相融合, 通过含有 Softmax 的全连接层连接到两个输出的类别。Dib 等人[101]使用恶意代码灰度图像作为静态特征, 行为信息字符串作为动态特征, 通过 CNN 网络和 Embedding+LSTM 网络分别提取图像模态和文本模态信息, 并将这两种模态的信息进行融合。他们也是最先将多维度特征融合分析方法应用到物联网恶意代码检测中的团队。

7.3 讨论

多维特征融合方法通过提取恶意代码静态和动态两个维度的特征进行分析, 使得动态分析和静态分析的优势互相补充, 劣势相互弥补, 因此在检测性能上较使用单一维度特征分析而言, 有着更好的检测效果。与此同时, 其缺点也应该得到充分的重视: ①该方法需要提取不同维度的特征, 因此特征提取的计算量会加大; ②使用不同维度的特征训练的模型有不同的检测效果, 当特征效果不对等时, 性能会大打折扣; ③静态和动态维度特征的耦合性强, 必须要同时输入两种维度特征才能进行检测。若仅有模型一种维度的特征, 则无法进行检测; ④如何设计特征融合方法, 也存在一定的难度。

在未来工作中, 以下几个方向值得我们进一步思考: ①恶意代码可以根据第5节和第6节的方法转换成恶意代码图像和类文本序列(API 序列), 借鉴图像模态和文本模态交互的任务, 如: 视觉问答[102-103]、图像文本检索[104]、图像标注[105]等任务, 以及相关模型, 设计使用恶意代码图像和类文本序列(API 序列)作为输入的多模态模型。②在部分情况下, 恶意代码难以同时提取两个维度的信息。在模型设计层面, 如何设计能在测试环节中进行维度解耦的模型, 使得用两个维度训练的模型, 在测试时能够用一个维度的信息进行测试, 并且保持相近的检测效果。③如何优化多维度特的融合方法, 也是值得研究的。

8 恶意代码检测中的攻防对抗

深度网络对抗的方法由 GoodFellow 首次提出[106], 使用基于梯度的攻击方法 FGSM(Fast Gradient Sign Method), 旨在通过对样本加入一些微小的扰动来规避分类器的检测, 如图 6 所示, 熊猫的图像加入微弱扰动, 就可以欺骗分类器将图片归类为长臂猿的图像。GAN(Generative Adversarial Network)生成方法也是由GoodFellow等人[107]提出, 模型可以对现有的数据进行无监督学习, 学习后的模型能生成出与原数据集相似且逼真的图片。

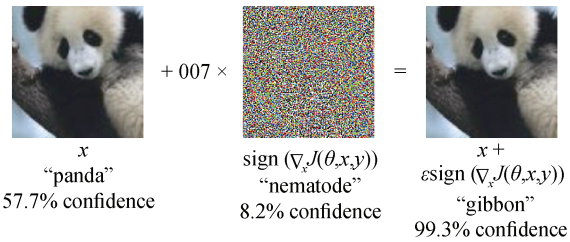

图 6 深度学习攻击方法示例[106]

Figure 6 Examples of deep learning attack methods

Pierazzi 等人[108]构建特征空间和攻击空间的关系, 并从可变换性、保留语义、预处理鲁棒性、合理性四个方面对关系进行约束。文章中对于 Windows 恶意代码的攻击分析只有一小部分, 但是对许多攻击操作的基本原理进行数学层面的分析和证明。对恶意代码进行轻微扰动需要遵循一些基本的原则: ①加入的扰动不能对恶意代码的关键功能造成影响; ②加入的扰动不能影响恶意代码的正常运行。否则, 即使规避了分类器的检测, 代码也损失了原来的攻击属性。目前, 绝大多数的攻击和对抗算法都是基于深度学习技术完成的。

8.1 梯度攻击的方法

受 GoodFellow 的启发, Grosse 等人[109]是第一个将生成对抗思想的方法运用到恶意代码生成上的人, 通过对原始代码进行一定程度的扰动以增强恶意代码的对抗性, 但在设计的过程中也存在一些难题, 比如: 增加扰动的过程中, 不能影响代码原有的功能, 通常只能往代码中加入额外的信息; GAN 的输入是图像, 是一种连续值; 而文章中的代码是以二进制形式进行表示, 是离散值, 这也导致 GAN 的思想运用在代码对抗上的效果并不是特别理想。

Kreuk 等人[110]使用基于 FGSM 的白盒攻击方法, 他们将离散的代码映射到连续的空间上, 对连续空间加入扰动后, 再从连续空间映射回离散的代码上。考虑到代码的完整性和可运行性, 冗余代码插入的位置也有所讲究, 在他们所提出的算法中, 插入的位置主要是 PE 头文件中的冗余位置以及代码末尾。

8.2 基于 GAN 的方法

应用GAN方法可以生成更具欺骗性的恶意代码伪样本。Hu 等人[111]使用基于生成对抗网络方法(MalGAN)设计对抗样本, 文中他们使用的数据为 API 特征, 通过往二进制代码中加入不相关的特征来构建对抗恶意样本。将对抗恶意样本和良性样本输入黑盒检测器, 即被攻击的分类器, 输入预测标签。将样本和对应的预测标签输入代理检测器中, 代理检测器尝试对黑盒检测器进行拟合。该方法可以在无需知道检测模型结构和梯度的情况下通过神经网络拟合检测器, 并生成对抗样本, 极大地提高恶意样本的逃避率。

Yuan 等人[112] 提出了基于 GAN 的黑盒对抗攻击方法 GAPGAN。通过在恶意代码尾部添加冗余内容以生成对抗代码。他们构建了二进制离散值与联系区间[–1,1]的映射, 借助此方式, 冗余内容可以像图像一样进行生成训练, 训练好的图像根据映射变回二进制离散值, 通过与黑盒检测器的交互来训练

生成器, 最终得到逃避率较高的恶意代码。

使用 GAN 方法提升模型的泛化能力的主要操作如图 7 所示, 收集原始的恶意代码图像数据, 接着用数据训练 GAN 模型, 训练好后的 GAN 模型能够生成高仿的恶意代码图像数据, 将这两种数据进行汇集进而对原始数据集扩容, 使用扩容后的数据集训练模型, 可以提升恶意代码检测模型的泛化能力。

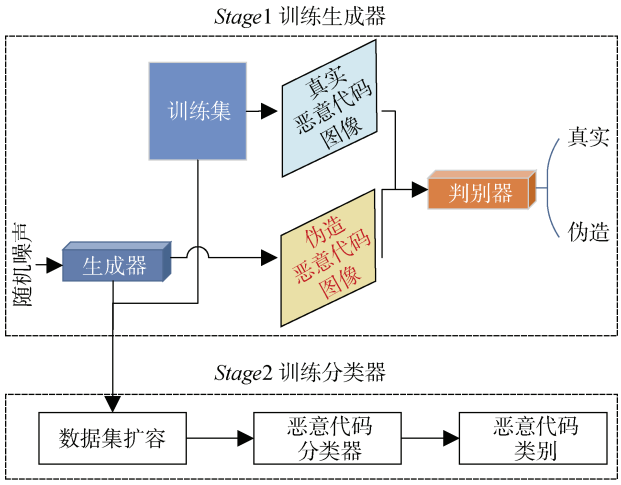

图 7　应用 GAN 提升模型的泛化能力

Figure 7　Applying GAN to improve the generalization ability of the model

Kim 等人[113] 通过转换生成动态网络(tGAN)生成更多的恶意代码图像和标签, 使用真实图片和生成图片训练模型。在下一阶段的工作中, Kim 等人[114]提出 LSC-GAN 模型, 使用变分自动编码器 (VAE)将数据投影到潜变量空间以进行特征提取, 并用生成器对其进行训练, 通过基于 VAE 生成的真实数据和修改数据学习恶意代码特征。在此基础上, Kim 等人[115]使用两种类型的数据训练模型。一种是恶意代码灰度图经过已训练的 GAN 模型编码的数据, 另一种是恶意代码 16 进制代码经过 CNN 层和 LSTM 层输出的数据, 将这两种数据输入进多层全连接层中, 最终输出各个恶意代码类别。

8.3　基于强化学习的方法

强化学习[116]由智能体(*Agent*)、环境(*Environment*)、状态(*State*)、动作(*Action*)、奖励(*Reward*)5 个部分组成, 通过智能体和环境相交互来对智能体进行训练。训练过程如图 8 所示, 智能体根据当前的环境和状态选择对应的动作, 完成动作后有相应的奖励并达到第 2 个状态, 奖励可能是正向也可能是负向, 根据奖励和第 2 个状态选择下一个需要完成的动作。和传统的深度学习相比, 强化学习在训练过程中对数据集的依赖程度会更低, 大部分所需的数据依靠与环境交互获得。

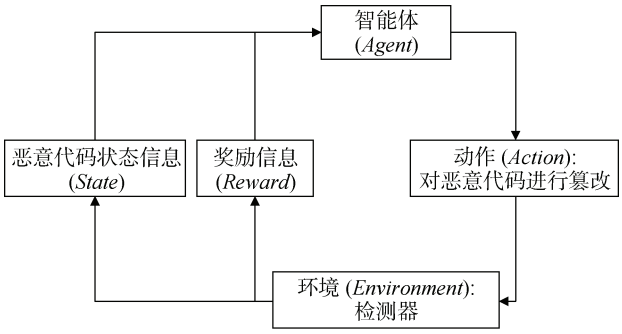

图 8　强化学习智能体训练流程图

Figure 8　Agent Training flowchart of reinforcement learning

Anderson 等人[117]使用基于强化学习的黑箱攻击方法, 通过对代码进行调整使得恶意代码逃避分类器的检测。他们设计一个智能体, 智能体通过对恶意代码进行改动(动作)以生成新的恶意代码, 将恶意代码输入进分类器, 分类器的分类效果决定了环境对智能体的反馈价值, 如果分类器分类错误, 则反馈价值为 10; 反之, 反馈价值为 0。将反馈价值的当前环境状态输入进智能体, 智能体根据信息进行下一步动作, 通过不断重复这个流程, 智能体能够针对恶意代码设计更好的调整方法以规避分类器的检测。GiBert 等人[118]设计了一种更为简单且有效的对抗样本生成方法, 仅需往恶意代码中加入无实际作用的“nop”指令以逃避分类器的检测。在他们设计的体系中, 奖励值为两个相邻时间步的损失值; 环境为汇编形式的恶意代码, 动作为添加“nop”的位置。智能体通过和环境进行交互, 采取动作对恶意代码进行改动, 即生成了新的环境状态, 接着由分类器进行判别, 分类器判别的过程中的结果作为奖励信息。智能体对新的环境状态和奖励信息进行学习, 在下一轮的迭代中能够采取更好的动作, 找到规避率更高的插入“nop”的方法。相较于在代码尾部或小节尾部添加内容的对抗方法, 该方法添加内容的位置有很强的随机性, 且插入的内容不影响恶意代码原始功能, 防止恶意代码预处理操作(如裁剪等)方法降低攻击方法的有效性。

强化学习一方面可以用于对抗样本的生成, 另一方面生成的对抗样本也可以用于增强检测模型的性能。Fang 等人[119]基于强化学习方法, 提出 RLAttackNet 网络, 使用双对偶 Q 价值网络[120]对动作策略进行创新。其中一个网络根据环境选择概率最大的动作 *Action*,另一个网络根据 *Action* 和环境计算出最大的 Q 价值, 智能体根据 *Action* 和 Q 价值, 做

出最终的动作。因此, 该模型能够找到更好的动作, 并在一定程度上抑制过估计现象。此外, 他们设计恶意代码检测模型 DetectNet, 使用 RLAttack 生成恶意代码对抗样本提升 DetectNet 的泛化能力。

8.4 讨论

基于GAN的方法是在计算机视觉方法的基础上发展而来的, 但自然图像与恶意代码图像存在着较大差异, 这些差异也会限制 GAN 方法在恶意代码图像处理中的使用范围和训练效果。自然图像是由连续数值表示的, 对连续数值加一些轻微的扰动, 基本不会影响图像的表征含义; 但是恶意代码不同, 恶意代码图像是由代码字节决定的, 代码字节本身就是离散型数据, 且各个代码字节之间存在着非常严格的逻辑关联, 无法像图像那样加入轻微的扰动。GAN 所生成的图像只能大致接近恶意代码的图像特征分布, 因此, 使用 GAN 生成的恶意代码图像可以用于提升分类模型的泛化性能, 但是并不具备真正意义上地生成恶意代码的能力。

基于强化学习的方法能够生成离散型扰动, 比如往代码中添加特定的指令、标记等, 并且添加的内容在注释、文件尾部等地方, 添加之后不影响代码的正常运行。因此相较于 GAN 方法而言, 强化学习的方法能够真正意义上生成对抗恶意代码样本。

生成与对抗方法可以用于恶意代码的生成与检测, 检测方法和生成方法在性能提升上相互促进, 呈螺旋式发展。生成方法所生成的对抗样本亦可以用于检测模型的训练, 以提升检测模型的检测效果和泛化性能。

近年来, 扩散生成模型[121]在图像生成领域取得了很大的突破。扩散模型主要包括两个过程: 向前扩散过程和反向扩散过程。向前扩散过程通过不断向图像添加高斯白噪声, 使其最终成为高斯白噪声; 反向扩散过程通过从高斯白噪声中不断减去噪声, 以恢复成原始图像。相较于 GAN 网络, 扩散模型的训练过程往往更加稳定。恶意代码图像在水平维度呈现很强的随机性, 这也表示我们需要更少的白噪声处理操作, 就能够完成向前和向后的扩散过程。因此, 如何设计面向恶意代码的扩散生成模型, 是后续值得思考的方向之一。

9 问题与挑战

目前的恶意代码检测方法在恶意代码数据集上表现出很好的效果。但是, 由恶意代码引起的网络安全问题并没有因此而得到明显好转。这是基于深度学习的恶意代码检测方法普遍存在的问题, 其成因也是多方面的, 既有模型结构本身的限制, 也有恶意代码不断变异所引起的逃避率的提升。

9.1 深度网络泛化性与鲁棒性

训练好的模型在已知的、特征分布相近的恶意代码上有着较好的检测效果, 但对于未知的、与训练数据分布特征有较大差异的恶意代码, 其检测效果往往并不理想。大部分研究方法在单一数据集上进行训练和测试, 其效果较好, 但鲜有研究提供跨数据集实验, 即用一个数据集训练模型, 用另一个同类型的其他数据集测试。设想当恶意代码被精心设计, 整体的特征分布与训练数据集中恶意代码的分布特征存在差异时, 恶意代码难以被检测出。因此, 泛化性能不高是基于深度学习方法的恶意代码检测方法的主要局限性之一。此外, 模型鲁棒性普遍不强, 如文章第 8 节所述, 一些模型通过攻击检测模型进行对抗训练, 能够学习到更好的噪声添加策略, 进一步提升恶意代码对抗样本的逃避率。

泛化性和鲁棒性不足问题, 在其他深度学习任务中也普遍存在, 除了较常见的数据集扩容和数据增强方法外, 恶意代码的攻防对抗为该问题的缓解有一定的启发。使用攻击端模型生成的对抗样本与正常样本一同训练模型能在一定程度上提升模型的泛化性和鲁棒性, 进而提升模型的检测性能。在自然语言任务中, 研究者们[122-123]通过对词嵌入层进行对抗攻击, 让模型学习到更为鲁棒的语义表达, 从而提升其在面对不同语境、语义变化时的检测性能。在此基础上, 我们是否可以使用梯度攻击法攻击基于自然语言处理的恶意代码检测方法的梯度, 以提升模型的泛化性、鲁棒性、检测效果, 这是值得进一步研究的问题。

9.2 概念漂移

随着时间的推进, 收集到的样本数据的统计特征信息以任意的方式和方向发生改变的现象, 称为概念漂移现象。概念漂移现象会导致使用现有的数据集训练好的模型, 无法对未来的数据进行检测。

Gibert 等人[2]认为在一些图像分类、文本分类等传统分类任务上, 历史样本和未来样本的数据分布差异不大, 因此并不需要考虑概念漂移的问题。但对于恶意代码检测任务而言, 恶意代码是在不断地进化和更新的, 经过多轮迭代后的代码在表征数据的分布上, 有可能会与训练集中的数据分布存在较大的差异。因此, 经过训练的模型难以对经过多轮迭代的恶意代码进行检测, 这种现象对模型的检测效果带来很大的影响。

根据目前的研究[124-125], 概念漂移主要包括四种

类型: 突变漂移、渐变漂移、增量漂移、概念重现, 但是恶意代码迭代的方向和强度具有很大的不确定性, 也很难归到上述任意一类。现有的概念漂移研究与恶意代码迭代的实际现象仍存在较大差距, 因此如何检测和缓解恶意代码在更新过程中的概念漂移现象仍是学界面临的主要难题之一。

针对此问题, 当我们获取到漂移样本的数据时, 用仅有的样本对模型进行有效的微调训练, 以应对漂移样本的检测, 这就需要模型对增量的数据有较好和较快的学习能力。增量学习[126]和小样本学习方法[127]比较贴合这方面的需要。增量学习通过对部分参数进行小幅度的修改, 实现在学习新样本的同时, 对旧样本的学习内容有所保留。小样本学习旨在学习恶意代码的特征表示, 即使对于在训练过程中未曾见过的类别, 小样本学习可以通过生成未知代码的特征表示, 并与已知代码的特征表示进行比较, 选择相似程度最高类别的作为最终预测的类别。此外, Zhang 等人[37]通过增强 API 序列调用图, 捕捉更高质量的特征信息, 对恶意代码更好地刻画, 实验证明该方法在对抗概念漂移现象和模型老化的问题上表现出很好的效果。

9.3 数据不平衡

数据不平衡是广泛存在于数据集中的问题, 即数据集中有些类别样本数量过多, 有些类别样本过少, 由此会导致模型训练不平衡。在样本少的类别上, 检测结果会相对差些。

如图 9 所示, 在主流的恶意代码数据集上, 样本数据不平衡问题仍十分显著。正样本收集容易, 但负样本的收集, 尤其是细分到具体类别或家族的收集工作尤为困难, 收集的过程中难免会存在数量不平衡和质量有限的问题。

针对数据不平衡问题, 最直接的方法有欠采样和过采样两种, 欠采样: 对于样本多的类别, 抽取部分数据构建训练集; 过采样, 对于样本少的类别, 重复抽取样本构建训练集。通过确定各个类别在训练中的权重, 也能在一定程度上缓解数据不平衡带来的影响: Yue 等人[128]使用带权重的 softmax 损失函数, 以缓解数据不平衡给模型训练带来的影响。此外, 部分研究者们使用智能优化算法重新确定各个类别的权重, Cui 等人[129]和 Cui 等人[130]分别使用 NSGA2 和蝙蝠优化算法确定类别的权重。

10 总结

本文对基于深度学习的恶意代码检测模型进行综述, 根据模型的输入数据类型、结构、训练方式等特点, 将现有的主流模型分为 5 个类别: 基于熵信息的方法、基于图的方法、基于计算机视觉的方法、基于自然语言处理的方法、基于多维度特征融合方法, 并对每个类别的通用框架进行分析, 优缺点汇总如表 4 所示。

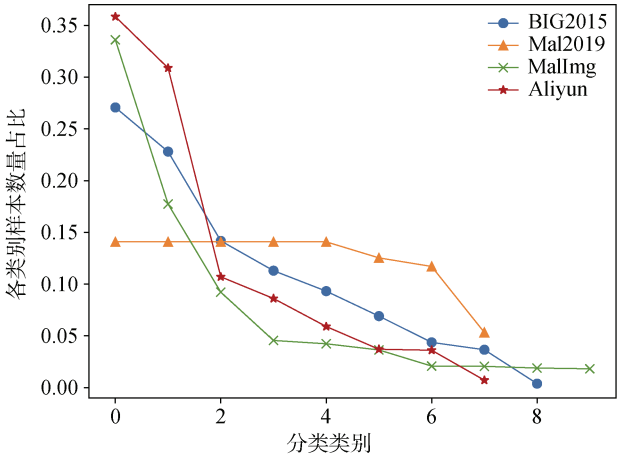

图 9 对 BIG2015 [64]、Mal2019[82]、MalImg[40]、Aliyun[91] 数据集中的各个类别样本占比进行统计, 按照占比大小由高到低排序并绘制折线图

Figure 9 samples proportion of each category in the BIG2015 [64], Mal2019 [82], MalImg [40], and Aliyun [91] datasets, sort them from high to low according to the proportion and draw a line chart

除了数据类别不平衡问题, 样本长度不平衡问题同样值得研究。不同的恶意代码长度并不相同, 对基于计算机视觉的方法而言, 即使恶意代码原始长度不同, 但是变成图像后可以进行缩放, 因此样本长度不平衡问题对基于计算机视觉的方法影响较小。但对于恶意代码动态分析而言有着很大的影响, 以 API 调用序列为例, 其 API 调用序列为离散特征, 不能像图像等连续特征那样进行自由的缩放。API 序列的长度范围在 $10^2 \sim 10^6$, 跨度非常大。序列信息随截断长度的变化情况如图 10 所示, 当截断长度大于序列长度时, 记为无损序列。在序列经过定长处理后, 非冗余值填充的内容占全部内容的比值记为有效信息占比。长度阈值过小, 则序列信息损失较多, 进而导致效果不高; 长度阈值过大, 有效信息占比下降, 计算时间和空间开销大, 因此如何更好地平衡每一条数据的长度显得至关重要。目前, 动态分析方法的设计或多或少源自于自然语言处理的思想, 如果从自然语言处理的角度思考这个问题, 可以借鉴文本摘要方法对长序列进行浓缩, 文本生成方法对短序列进行延拓, 缓解样本长度不平衡对动态分析方法的影响。

表 4 各个类别方法的优缺点简要汇总

Table 4 A brief summary of each category's advantages and disadvantages

类别	优点	缺点
基于熵信息的方法	特征获取过程相对简单; 鲁棒性强, 能够有效防止无效代码插入、代码位置变换、加密与包装等混淆技术对检测器的影响。	只关注代码的统计信息, 缺少对代码语义信息的关注。
基于图的方法	图方法可以更好地表示代码的结构和行为, 挖掘代码中的复杂模式和关系。	图结构构建过程复杂, 表示维度高。
基于计算机视觉的方法	操作简单, 只需将特征按照一定的规则转化成图片, 之后使用深度学习方法进行端到端的学习; 在同一数据集内进行训练和测试, 有着较高的检测率。	对于经过加壳、混淆等操作的恶意代码时, 该方法的检测效果往往并不理想; 此外, 该方法的解释性差。
基于自然语言处理的方法	相较于计算机视觉方法而言, 该方法能够大幅降低加壳、混淆等方法对检测器的影响; 此外, 该方法有着较强的解释性。	数据获取工作量大。需要将每个恶意代码样本放进沙箱中运行, 以记录其 API 调用序列, 这个过程需要花费大量的时间。
基于多维度特征融合的方法	通过静态和动态两个维度对恶意代码进行分析, 能够更好地对恶意代码的特征进行刻画, 在一定程度上提升模型的检测性能。	特征提取工作量大; 当静态特征和动态特征所表现的性能不对等时, 模型的检测性能将会降低; 检测时需要同时输入 2 种维度的特征, 如果仅有一种维度的特征时, 检测效果会大幅降低, 甚至无法进行检测。

作为检测模型的对立面, 攻击模型通过对恶意代码进行修改, 企图逃过检测。但我们要辩证地看待其使用和研究的价值。一方面, 通过攻击模型的对抗测试, 能测出检测模型存在的缺陷; 另一方面, 通过攻击模型生成对抗样本, 将对抗样本和真实样本一同用于检测模型的训练能够提升检测模型的泛化能力。

得益于深度学习模型在图像和文本分类任务上面有着极好的性能, 在恶意代码领域, 研究者们将恶意代码特征通过一定的方法将其转化成恶意代码图像和类文本序列, 进而使用计算机视觉和自然语言处理的方法进行分析检测。此外, 本文对恶意代码图像和 API 调用序列, 与自然图像和自然文本进行分析, 比较其中同异, 以对后面在恶意代码特征设计相关工作上有所启发。

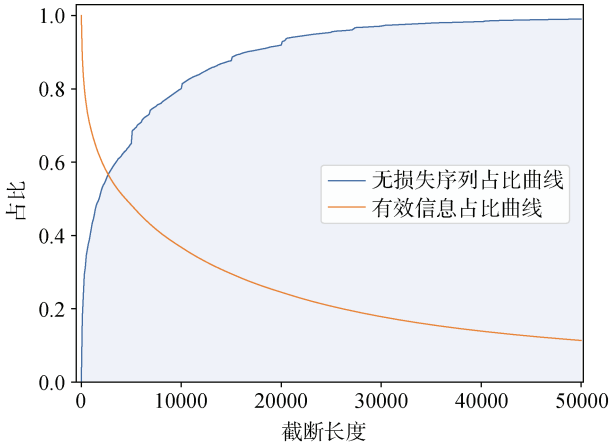

图 10 无损序列占比和有效信息占比随截断长度变化的曲线图

Figure 10 relationship of the lossless sequences proportion and the valid information proportion to the truncation length

恶意代码检测方法在深度学习的大环境和成熟框架的支持下, 衍生出以深度学习为基础的不同方法类别和发展路径。这些类别、路径虽有不同, 但也在相互促进和相互补充, 即使发展过程中会存在一些缺陷和不足, 但基于深度学习的技术仍是目前恶意代码检测的主流方法。

目前, 基于深度学习方法的模型对已知的代码有很好的检测效果, 模型性能也在不断提升, 但有些问题仍没能很好解决。如何提升模型的泛化性能和鲁棒性能, 以抵御新型恶意代码和对抗样本的攻击; 检测细粒度能否进一步提升, 即在分类的基础上, 能否进一步对恶意代码的特征片段进行定位; 目前主流数据集样本不平衡现象严重, 能否创建一个质量更好、类别更全更平衡的数据集, 能否设计更好的算法减轻类别不平衡、样本长度不平衡对训练过程带来的影响; 恶意代码更新迭代速度快, 所带来的概念漂移现象是否能更好地被检测出和抑制。这些学术问题将引导今后恶意代码检测方法的研究方向。

参考文献

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

[2] Gibert D, Mateu C, Planes J. The Rise of Machine Learning for Detection and Classification of Malware: Research Developments, Trends and Challenges[J]. *Journal of Network and Computer Applications*, 2020, 153: 102526.

[3] Malware - Wikipedia. https://en.wikipedia.org/wiki/Malware. 2023.

[4] Gandotra E, Bansal D, Sofat S. Malware Analysis and Classification: A Survey[J]. *Journal of Information Security*, 2014, 5(2): 56-64.

[5] Idika N, Mathur A P. A survey of malware detection techniques[J]. *Purdue University*, 2007, 48(2): 32-46.

[6] Mitchell T M. Machine learning[M]. 1. McGraw-hill New York, 2007.

[7] Sibi Chakkaravarthy S, Sangeetha D, Vaidehi V. A Survey on Malware Analysis and Mitigation Techniques[J]. *Computer Science Review*, 2019, 32: 1-23.

[8] Sihwail R, Omar K, Zainol Ariffin K A. A Survey on Malware Analysis Techniques: Static, Dynamic, Hybrid and Memory Analysis[J]. *International Journal on Advanced Science, Engineering and Information Technology*, 2018, 8: 1662-1671.

[9] Ucci D, Aniello L, Baldoni R. Survey of Machine Learning Techniques for Malware Analysis[J]. *Computers & Security*, 2019, 81: 123-147.

[10] Long T, Gao Q, Xu L L, et al. A Survey on Adversarial Attacks in Computer Vision: Taxonomy, Visualization and Future Directions[J]. *Computers & Security*, 2022, 121: 102847.

[11] Qiu J Y, Zhang J, Luo W, et al. A Survey of Android Malware Detection with Deep Neural Models[J]. *ACM Computing Surveys*, 2020, 53(6): 1-36.

[12] Wang D X, Chen T, Zhang Z, et al. A Survey ofAndroid Malware Detection Based onDeep Learning[C]. *Machine Learning for Cyber Security*, 2023: 228-242.

[13] Ali R, Ali A, Iqbal F, et al. Deep Learning Methods for Malware and Intrusion Detection: A Systematic Literature Review[J]. *Security and Communication Networks*, 2022, 2022(1): 2959222.

[14] Sahin M, Bahtiyar S, Sahin M, et al. A Survey on Malware Detection with Deep Learning[C]. *13th International Conference on Security of Information and Networks*, 2021: 1-6.

[15] Berman D S, Buczak A L, Chavis J S, et al. A Survey of Deep Learning Methods for Cyber Security[J]. *Information*, 2019, 10(4): 122.

[16] Moser A, Kruegel C, Kirda E. Limits of Static Analysis for Malware Detection[C]. *Twenty-Third Annual Computer Security Applications Conference*, 2007: 421-430.

[17] Willems C, Holz T, Freiling F. Toward Automated Dynamic Malware Analysis Using CWSandbox[J]. *IEEE Security and Privacy Magazine*, 2007, 5(2): 32-39.

[18] Anderson B, Storlie C, Lane T. Improving Malware Classification: Bridging the Static/Dynamic Gap[C]. *The 5th ACM Workshop on Security and Artificial Intelligence*, 2012: 3-14.

[19] Santos I, Brezo F, Ugarte-Pedrero X, et al. Opcode Sequences as Representation of Executables for Data-Mining-Based Unknown Malware Detection[J]. *Information Sciences*, 2013, 231: 64-82.

[20] Santos I, Brezo F, Nieves J, et al. Idea: Opcode-Sequence-Based Malware Detection[C]. *Engineering Secure Software and Systems*, 2010: 35-43.

[21] Moskovitch R, Feher C, Tzachar N, et al. Unknown Malcode Detection Using OPCODE Representation[C]. *Intelligence and Security Informatics*, 2008: 204-215.

[22] Shijo P V, Salim A. Integrated Static and Dynamic Analysis for Malware Detection[J]. *Procedia Computer Science*, 2015, 46: 804-811.

[23] Islam R, Tian R H, Batten L M, et al. Classification of Malware Based on Integrated Static and Dynamic Features[J]. *Journal of Network and Computer Applications*, 2013, 36(2): 646-656.

[24] Santos I, Devesa J, Brezo F, et al. OPEM: A Static-Dynamic Approach for Machine-Learning-Based Malware Detection[C]. *International Joint Conference CISIS'12-ICEUTE'12-SOCO'12 Special Sessions*, 2013: 271-280.

[25] Mosli R, Li R, Yuan B, et al. A Behavior-Based Approach for Malware Detection[C]. *Advances in Digital Forensics XIII. Cham: Springer International Publishing*, 2017: 187-201.

[26] Zhong W, Gu F. A Multi-Level Deep Learning System for Malware Detection[J]. *Expert Systems with Applications*, 2019, 133: 151-162.

[27] Lyda R, Hamrock J. Using Entropy Analysis to Find Encrypted and Packed Malware[J]. *IEEE Security and Privacy*, 2007, 5(2): 40-45.

[28] Bat-Erdene M, Park H, Li H Z, et al. Entropy Analysis to Classify Unknown Packing Algorithms for Malware Detection[J]. *International Journal of Information Security*, 2017, 16(3): 227-248.

[29] Gibert D, Planes J, Mateu C, et al. Fusing Feature Engineering and Deep Learning: A Case Study for Malware Classification[J]. *Expert Systems with Applications*, 2022, 207: 117957.

[30] Gibert D, Mateu C, Planes J, et al. Classification of Malware by Using Structural Entropy on Convolutional Neural Networks[C]. *The Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, 2018: 7759-7764.

[31] Vu D L, Nguyen T K, Nguyen T V, et al. HIT4Mal: Hybrid Image Transformation for Malware Classification[J]. *Transactions on Emerging Telecommunications Technologies*, 2020, 31(11): e3789.

[32] Li C, Cheng Z J, Zhu H, et al. DMalNet: Dynamic Malware Analysis Based on API Feature Engineering and Graph Learning[J]. *Computers & Security*, 2022, 122: 102872.

[33] Kipf T N, Welling M. Semi-Supervised Classification with Graph Convolutional Networks[EB/OL]. 2016: 1609.02907. https://arxiv.org/abs/1609.02907v4.

[34] Veličković P, Cucurull G, Casanova A, et al. Graph Attention Networks[EB/OL]. 2017: 1710.10903. https://arxiv.org/abs/1710.10903v3.

[35] Pei X J, Yu L, Tian S W. AMalNet: A Deep Learning Framework Based on Graph Convolutional Networks for Malware Detection[J]. *Computers & Security*, 2020, 93: 101792.

[36] Li S, Li W Q, Cook C, et al. Independently Recurrent Neural Network (IndRNN): Building a Longer and Deeper RNN[C]. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018: 5457-5466.

[37] Zhang X H, Zhang Y, Zhong M, et al. Enhancing State-of-the-Art Classifiers with API Semantics to Detect Evolved Android Malware[C]. *The 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020: 757-770.

[38] Wang Q, Hassan W U, Li D, et al. You Are What You Do: Hunting

Stealthy Malware *via* Data Provenance Analysis[C]. *Proceedings 2020 Network and Distributed System Security Symposium*, 2020.

[39] Wang S, Wang Z L, Zhou T, et al. THREATRACE: Detecting and Tracing Host-Based Threats in Node Level through Provenance Graph Learning[J]. *IEEE Transactions on Information Forensics and Security*, 2022, 17: 3972-3987.

[40] Nataraj L, Karthikeyan S, Jacob G, et al. Malware Images: Visualization and Automatic Classification[C]. *The 8th International Symposium on Visualization for Cyber Security*, 2011: 1-7.

[41] Ni S, Qian Q, Zhang R. Malware Identification Using Visualization Images and Deep Learning[J]. *Computers & Security*, 2018, 77: 871-885.

[42] Makandar A, Patrot A. Malware Analysis and Classification Using Artificial Neural Network[C]. *2015 International Conference on Trends in Automation*, *Communications and Computing Technology*, 2015: 1-6.

[43] Zhao Y T, Xu C Y, Bo B, et al. MalDeep: A Deep Learning Classification Framework Against Malware Variants Based on Texture Visualization[J]. *Security and Communication Networks*, 2019, 2019(1): 4895984.

[44] Gibert D, Mateu C, Planes J, et al. Using Convolutional Neural Networks for Classification of Malware Represented as Images[J]. *Journal of Computer Virology and Hacking Techniques*, 2019, 15(1): 15-28.

[45] Kalash M, Rochan M, Mohammed N, et al. Malware Classification with Deep Convolutional Neural Networks[C]. *2018 9th IFIP International Conference on New Technologies*, *Mobility and Security*, 2018: 1-5.

[46] Mourtaji Y, Bouhorma M, Alghazzawi D, et al. Intelligent Framework for Malware Detection with Convolutional Neural Network[C]. *The 2nd International Conference on Networking*, *Information Systems & Security*, 2019: 1-6.

[47] Vasan D, Alazab M, Wassan S, et al. Image-Based Malware Classification Using Ensemble of CNN Architectures (IMCEC)[J]. *Computers & Security*, 2020, 92: 101748.

[48] Deng J, Dong W, Socher R, et al. ImageNet: A Large-Scale Hierarchical Image Database[C]. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009: 248-255.

[49] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition[EB/OL]. 2014: 1409.1556. https://arxiv.org/abs/1409.1556v6.

[50] He K M, Zhang X Y, Ren S Q, et al. Deep Residual Learning for Image Recognition[C]. *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016: 770-778.

[51] Parihar A S, Kumar S, Khosla S. S-DCNN: Stacked Deep Convolutional Neural Networks for Malware Classification[J]. *Multimedia Tools and Applications*, 2022, 81(21): 30997-31015.

[52] Chollet F. Xception: Deep Learning with Depthwise Separable Convolutions[C]. *2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2017: 1800-1807.

[53] Tan M X, Le Q V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks[EB/OL]. 2019: 1905.11946. https://arxiv.org/abs/1905.11946v5.

[54] Nguyen M H, Le Nguyen D, Nguyen X M, et al. Auto-Detection of Sophisticated Malware Using Lazy-Binding Control Flow Graph and Deep Learning[J]. *Computers & Security*, 2018, 76: 128-155.

[55] Yuan B G, Wang J F, Liu D, et al. Byte-Level Malware Classification Based on Markov Images and Deep Learning[J]. *Computers & Security*, 2020, 92: 101740.

[56] Tang M D, Qian Q. Dynamic API Call Sequence Visualisation for Malware Classification[J]. *IET Information Security*, 2019, 13(4): 367-377.

[57] Tobiyama S, Yamaguchi Y, Shimada H, et al. Malware Detection with Deep Neural Network Using Process Behavior[C]. *2016 IEEE 40th Annual Computer Software and Applications Conference*, 2016: 577-582.

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

[59] Devlin J, Chang M W, Lee K, et al. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding[EB/OL]. 2018: 1810.04805. https://arxiv.org/abs/1810.04805v2.

[60] Improving language understanding by generative pre-training. https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf. 2023.

[61] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale[J]. *arXiv preprint arXiv:2010.11929*, 2020.

[62] Yakura H, Shinozaki S, Nishimura R, et al. Neural Malware Analysis with Attention Mechanism[J]. *Computers & Security*, 2019, 87: 101592.

[63] Seneviratne S, Shariffdeen R, Rasnayaka S, et al. Self-Supervised Vision Transformers for Malware Detection[J]. *IEEE Access*, 2022, 10: 103121-103135.

[64] Ronen R, Radu M, Feuerstein C, et al. Microsoft Malware Classification Challenge[EB/OL]. 2018: 1802.10135. https://arxiv.org/abs/1802.10135v1.

[65] Howard A, Sandler M, Chen B, et al. Searching for MobileNetV3[C]. *2019 IEEE/CVF International Conference on Computer Vision*, 2019: 1314-1324.

[66] Cuckoo Sandbox - Automated Malware Analysis. https://cuckoosandbox.org/. 2023.

[67] CAPE Sandbox. https://capesandbox.com/. 2023.

[68] Cavnar W B, Trenkle J M. N-gram-based text categorization[C]. *the 3rd Annual Symposium on Document Analysis and Information Retrieval*, 1994.

[69] Yoon K. Convolutional Neural Networks for Sentence Classification[C]. *Conference on Empirical Methods in Natural Language Processing*, 2023.

[70] Qin B, Wang Y L, Ma C C. API Call Based Ransomware Dynamic Detection Approach Using TextCNN[C]. *2020 International Conference on Big Data*, *Artificial Intelligence and Internet of Things Engineering*, 2020: 162-166.

[71] Mikolov T, Chen K, Corrado G, et al. Efficient Estimation of Word Representations in Vector Space[EB/OL]. 2013: 1301.3781. https://arxiv.org/abs/1301.3781v3.

[72] K B S, N B, Prithvikiran. Malware Classification Using Deep Learning Methods[C]. *2023 3rd International Conference on*

Smart Data Intelligence, 2023: 278-281.

[73] Raff E, Barker J, Sylvester J, et al. Malware Detection by Eating a Whole EXE[EB/OL]. 2017: 1710.09435. https://arxiv.org/abs/1710.09435v1.

[74] Krcál M, Švec O, Bálek M, et al. Deep Convolutional Malware Classifiers Can Learn from Raw Executables and Labels Only[C]. *International Conference on Learning Representations*, 2019.

[75] Zhang B, Xiao W T, Xiao X, et al. Ransomware Classification Using Patch-Based CNN and Self-Attention Network on Embedded N-Grams of Opcodes[J]. *Future Generation Computer Systems*, 2020, 110: 708-720.

[76] Sung Y, Jang S, Jeong Y S, et al. Malware Classification Algorithm Using Advanced Word2Vec-Based Bi-LSTM for Ground Control Stations[J]. *Computer Communications*, 2020, 153: 342-348.

[77] Bojanowski P, Grave E, Joulin A, et al. Enriching Word Vectors with Subword Information[J]. *Transactions of the Association for Computational Linguistics*, 2017, 5: 135-146.

[78] Joulin A, Grave E, Bojanowski P, et al. Bag of Tricks for Efficient Text Classification[EB/OL]. 2016: 1607.01759. https://arxiv.org/abs/1607.01759v3.

[79] Pascanu R, Stokes J W, Sanossian H, et al. Malware Classification with Recurrent Networks[C]. *2015 IEEE International Conference on Acoustics*, *Speech and Signal Processing*, 2015: 1916-1920.

[80] Jaeger H. Echo State Network[J]. *Scholarpedia*, 2007, 2(9): 2330.

[81] Maniath S, Ashok A, Poornachandran P, et al. Deep Learning LSTM Based Ransomware Detection[C]. *2017 Recent Developments in Control*, *Automation & Power Engineering*, 2017: 442-446.

[82] Catak F O, Yazı A F, Elezaj O, et al. Deep Learning Based Sequential Model for Malware Analysis Using Windows Exe API Calls[J]. *PeerJ Computer Science*, 2020, 6: e285.

[83] Athiwaratkun B, Stokes J W. Malware Classification with LSTM and GRU Language Models and a Character-Level CNN[C]. *2017 IEEE International Conference on Acoustics*, *Speech and Signal Processing*, 2017: 2482-2486.

[84] Hochreiter S, Schmidhuber J. Long Short-Term Memory[J]. *Neural Computation*, 1997, 9(8): 1735-1780.

[85] Chung J, Gulcehre C, Cho K, et al. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling[EB/OL]. 2014: 1412.3555. https://arxiv.org/abs/1412.3555v1.

[86] Kolosnjaji B, Zarras A, Webster G, et al. Deep Learning for Classification of Malware System Call Sequences[C]. *AI 2016: Advances in Artificial Intelligence*, 2016: 137-149.

[87] Li C, Lv Q J, Li N, et al. A Novel Deep Framework for Dynamic Malware Detection Based on API Sequence Intrinsic Features[J]. *Computers & Security*, 2022, 116: 102686.

[88] 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.

[89] Xu Z F, Fang X J, Yang G M. Malbert: A Novel Pre-Training Method for Malware Detection[J]. *Computers & Security*, 2021, 111: 102458.

[90] Huang Y T, Sun Y S, Chen M C. TagSeq: Malicious Behavior Discovery Using Dynamic Analysis[J]. *PLoS ONE*, 2022, 17(5): e0263644.

[91] Alibaba Cloud Malware Detection Based On Behaviors. https://tianchi.aliyun.com/getStart/information.htm?raceId=231694. 2018.

[92] Bengio Y, Simard P, Frasconi P. Learning Long-Term Dependencies with Gradient Descent Is Difficult[J]. *IEEE Transactions on Neural Networks*, 1994, 5(2): 157-166.

[93] Schuster M, Paliwal K K. Bidirectional Recurrent Neural Networks[J]. *IEEE Transactions on Signal Processing*, 1997, 45(11): 2673-2681.

[94] Huang W Y, Stokes J W. MtNet: A Multi-Task Neural Network for Dynamic Malware Classification[C]. *Detection of Intrusions and Malware, and Vulnerability Assessment*, 2016: 399-418.

[95] THUCTC: An Efficient Chinese Text Classifier. http://thuctc.thunlp.org/. 2016.

[96] Radford A, Kim J W, Hallacy C, et al. Learning transferable visual models from natural language supervision[C]. *International Conference on Machine Learning*, 2021: 8748-8763.

[97] Xu L F, Zhang D P, Jayasena N, et al. HADM: Hybrid Analysis for Detection of Malware[C]. *Proceedings of SAI Intelligent Systems Conference*, 2018: 702-724.

[98] Gibert D, Mateu C, Planes J. HYDRA: A Multimodal Deep Learning Framework for Malware Classification[J]. *Computers & Security*, 2020, 95: 101873.

[99] Huang X, Ma L, Yang W Y, et al. A Method for Windows Malware Detection Based on Deep Learning[J]. *Journal of Signal Processing Systems*, 2021, 93(2): 265-273.

[100] Zhang J X, Qin Z, Yin H, et al. A Feature-Hybrid Malware Variants Detection Using CNN Based Opcode Embedding and BPNN Based API Embedding[J]. *Computers & Security*, 2019, 84: 376-392.

[101] Dib M, Torabi S, Bou-Harb E, et al. A Multi-Dimensional Deep Learning Framework for IoT Malware Classification and Family Attribution[J]. *IEEE Transactions on Network and Service Management*, 2021, 18(2): 1165-1177.

[102] Yang Z C, He X D, Gao J F, et al. Stacked Attention Networks for Image Question Answering[C]. *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016: 21-29.

[103] Fukui A, Park D H, Yang D, et al. Multimodal Compact Bilinear Pooling for Visual Question Answering and Visual Grounding[EB/OL]. 2016: 1606.01847. https://arxiv.org/abs/1606.01847v3.

[104] Frome A, Corrado G S, Shlens J, et al. DeViSE: A Deep Visual-Semantic Embedding Model[J]. *Advances in Neural Information Processing Systems*, 2013: 26.

[105] Niu Z X, Zhou M, Wang L, et al. Hierarchical Multimodal LSTM for Dense Visual-Semantic Embedding[C]. *2017 IEEE International Conference on Computer Vision*, 2017: 1899-1907.

[106] Goodfellow I J, Shlens J, Szegedy C, et al. Explaining and Harnessing Adversarial Examples[EB/OL]. 2014: 1412.6572. https://arxiv.org/abs/1412.6572v3.

[107] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative Adversarial Networks[J]. *Communications of the ACM*, 2020, 63(11): 139-144.

[108] Pierazzi F, Pendlebury F, Cortellazzi J, et al. Intriguing Properties of Adversarial ML Attacks in the Problem Space[C]. *2020 IEEE Symposium on Security and Privacy*, 2020: 1332-1349.

[109] Grosse K, Papernot N, Manoharan P, et al. Adversarial Perturbations Against Deep Neural Networks for Malware Classification[EB/OL]. 2016: 1606.04435. https://arxiv.org/abs/1606.04435v2.

[110] Kreuk F, Barak A, Aviv-Reuven S, et al. Deceiving End-to-End Deep Learning Malware Detectors Using Adversarial Examples [EB/OL]. 2018: 1802.04528. https://arxiv.org/abs/1802.04528v3.

[111] Hu W W, Tan Y. Generating Adversarial Malware Examples ForBlack-Box Attacks Based onGAN[C]. *Data Mining and Big Data*, 2022: 409-423.

[112] Yuan J, Zhou S, Lin L, et al. Black-box adversarial attacks against deep learning based malware binaries detection with GAN[C]. *24th European Conference on Artificial Intelligence*, 2020: 2536-2542.

[113] Kim J Y, Bu S J, Cho S B. Malware Detection Using Deep Transferred Generative Adversarial Networks[C]. *Neural Information Processing*, 2017: 556-564.

[114] Kim J Y, Cho S B. Detecting Intrusive Malware with a Hybrid Generative Deep Learning Model[C]. *Intelligent Data Engineering and Automated Learning – IDEAL 2018*, 2018: 499-507.

[115] Kim J Y, Cho S B. Obfuscated Malware Detection Using Deep Generative Model Based on Global/Local Features[J]. *Computers & Security*, 2022, 112: 102501.

[116] Mnih V, Kavukcuoglu K, Silver D, et al. Human-Level Control through Deep Reinforcement Learning[J]. *Nature*, 2015, 518(7540): 529-533.

[117] Anderson H S, Kharkar A, Filar B, et al. Evading machine learning malware detection[J]. *Black Hat*, 2017, 2017.

[118] Gibert D, Fredrikson M, Mateu C, et al. Enhancing the Insertion of NOP Instructions to Obfuscate Malware *via* Deep Reinforcement Learning[J]. *Computers & Security*, 2022, 113: 102543.

[119] Fang Y, Zeng Y T, Li B B, et al. DeepDetectNet vs RLAttackNet: An Adversarial Method to Improve Deep Learning-Based Static Malware Detection Model[J]. *PLoS ONE*, 2020, 15(4): e0231626.

[120] van Hasselt H, Guez A, Silver D, et al. Deep Reinforcement Learning with Double Q-Learning[C]. *The Thirtieth AAAI Conference on Artificial Intelligence*, 2016: 2094-2100.

[121] Ho J, Jain A, Abbeel P, et al. Denoising Diffusion Probabilistic Models[C]. *The 34th International Conference on Neural Information Processing Systems*, 2020: 6840-6851.

[122] Miyato T, Dai A M, Goodfellow I. Adversarial Training Methods for Semi-Supervised Text Classification[EB/OL]. 2016: 1605.07725. https://arxiv.org/abs/1605.07725v4.

[123] Sato M, Suzuki J, Shindo H, et al. Interpretable Adversarial Perturbation in Input Embedding Space for Text[EB/OL]. 2018: 1805.02917. https://arxiv.org/abs/1805.02917v1.

[124] Lu J, Liu A J, Dong F, et al. Learning under Concept Drift: A Review[J]. *IEEE Transactions on Knowledge and Data Engineering*, 2019, 31(12): 2346-2363.

[125] Lu N, Zhang G Q, Lu J. Concept Drift Detection *via* Competence Models[J]. *Artificial Intelligence*, 2014, 209: 11-28.

[126] R A R, R D P. Methods for Incremental Learning: A Survey[J]. *International Journal of Data Mining & Knowledge Management Process*, 2013, 3(4): 119-125.

[127] Koch G, Zemel R, Salakhutdinov R. Siamese neural networks for one-shot image recognition[C]. *ICML Deep Learning Workshop*, 2015.

[128] Yue S Q, Wang T Y. Imbalanced Malware Images Classification: A CNN Based Approach[EB/OL]. 2017: 1708.08042. https://arxiv.org/abs/1708.08042v2.

[129] Cui Z H, Du L, Wang P H, et al. Malicious Code Detection Based on CNNS and Multi-Objective Algorithm[J]. *Journal of Parallel and Distributed Computing*, 2019, 129: 50-58.

[130] Cui Z H, Xue F, Cai X J, et al. Detection of Malicious Code Variants Based on Deep Learning[J]. *IEEE Transactions on Industrial Informatics*, 2018, 14(7): 3187-3196.

严沛 于 2022 年在华南师范大学计算机科学与技术专业获得学士学位。现在在深圳大学信息与通信工程专业攻读硕士学位。研究领域为: 网络信息安全, 研究兴趣包括: 恶意代码检测、自然语言处理。Email: yanpei2022@email.szu.edu.cn

谭舜泉 于 2007 年在中山大学计算机软件与理论专业获得博士学位。现任深圳大学计算机与软件学院副教授, 深圳市媒体信息内容安全重点实验室副主任。研究领域为多媒体信息安全。研究兴趣包括多媒体取证, 机器学习及深度学习在多媒体信息安全领域的应用。Email: tansq@szu.edu.cn

黄继武 于 1998 年在中国科学院自动化所模式识别与智能系统专业获得博士学位。现为深圳大学电子与信息工程学院特聘教授、IEEE Fellow。广东省网络与信息安全产学研创新联盟理事长。研究兴趣为多媒体取证与安全、信息隐藏。Email: jwhuang@szu.edu.cn