

基于注意力与门控机制的多特征融合恶意软件检测方法

陈仲元, 张建标

(北京工业大学信息学部计算机学院, 北京 100124)

摘要: 随着网络技术的飞速发展, 恶意软件及其变种的数量不断增加, 这使得恶意软件的检测成为网络安全领域面临的一大挑战。然而, 现有的单一特征恶意软件检测方法在样本信息的表示上存在不足, 而对于采用多特征的检测方法, 它们在特征融合方面存在局限, 未能有效地学习和理解特征内部及特征间的复杂关联, 这些问题都会导致检测效果不佳。提出了一种基于多模态特征融合的恶意软件检测方法——MFAGM。通过处理数据集的.asm和.bytes文件, 成功提取了两种类型的3种关键特征(操作码统计序列、API序列和灰度图像特征), 实现了从多个角度全面地表征样本信息。为了更好地融合这些多模态特征, 设计了一个特征融合模块SA-JGmu。该模块不仅采用自注意力机制捕获特征之间的内部依赖关系, 还利用门控机制增强了不同特征的交互性, 并巧妙地引入了权重跳跃连接以进一步优化模型的表示能力。最终, 基于MMCC(Microsoft malware classification challenge)数据集的实验结果显示, MFAGM在恶意软件检测任务上与其他方法相比, 达到了更高的准确率和F1分数。

关键词: 恶意软件检测; 深度学习; 特征融合; 多模态学习; 静态分析

中图分类号: TP309.5

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2024002

Multi-feature fusion malware detection method based on attention and gating mechanisms

CHEN Zhongyuan, ZHANG Jianbiao

School of Computer Science, Department of Information Science, Beijing University of Technology, Beijing 100124, China

Abstract: With the rapid development of network technology, the number and variety of malware have been increasing, posing a significant challenge in the field of network security. However, existing single-feature malware detection methods have proven inadequate in representing sample information effectively. Moreover, multi-feature detection approaches also face limitations in feature fusion, resulting in an inability to learn and comprehend the complex relationships within and between features. These limitations ultimately lead to subpar detection results. To address these issues, a malware detection method called MFAGM was proposed, which focused on multimodal feature fusion. By processing the .asm and .bytes files of the dataset, three key features belonging to two types (opcode statistics sequences, API sequences, and

收稿日期: 2023-11-21; 修回日期: 2024-01-03

通信作者: 张建标, zjb@bjut.edu.cn

基金项目: 北京市自然科学基金(M21039)

Foundation Item: Beijing Natural Science Foundation (M21039)

引用格式: 陈仲元, 张建标. 基于注意力与门控机制的多特征融合恶意软件检测方法[J]. 网络与信息安全学报, 2024, 10(1): 123-135.

Citation Format: CHEN Z Y, ZHANG J B. MFAGM: multi-feature fusion malware detection method based on attention and gating mechanisms[J]. Chinese Journal of Network and Information Security, 2024, 10(1): 123-135.

grey-scale image features) were successfully extracted. This comprehensive characterization of sample information from multiple perspectives aimed to improve detection accuracy. In order to enhance the fusion of these multimodal features, a feature fusion module called SA-JGmu was designed. This module utilized the self-attention mechanism to capture internal dependencies between features. It also leveraged the gating mechanism to enhance interactivity among different features. Additionally, weight-jumping links were introduced to further optimize the representational capabilities of the model. Experimental results on the Microsoft malware classification challenge dataset demonstrate that MFAGM achieves higher accuracy and $F1$ scores compared to other methods in the task of malware detection.

Keywords: malware detection, deep learning, feature fusion, multimodal learning, static analysis

0 引言

在过去几十年里,关于个人计算机和网络中恶意软件的报告数量持续增加。BlackBerry 在 2023 年 11 月发布的报告^[1]显示,其产品在 2023 年 6 月到 8 月平均每天可以监测到 4 237 个新型恶意软件样本,共检测到 381 340 个。恶意软件不仅会对设备造成损害,还会窃取个人数据并导致系统混乱,对用户构成严重威胁。因此,检测恶意软件对于防止用户受到进一步威胁^[2],以及对系统和网络安全来说都至关重要^[3]。

恶意软件检测主要包含动态分析和静态分析两种方法。动态分析通过在受控环境(如虚拟机、模拟器或沙箱)中观察应用程序的行为来揭示其实际操作,如文件访问、应用程序接口(API, application program interface)调用^[4-6]等。这些行为被研究人员记录并用于生成报告,研究人员可以通过分析这些报告中的特征来有效地对恶意软件进行检测^[7-10]。尽管动态分析能够揭示程序的真实行为、提高恶意软件检测的准确性,并能够抵御加密和混淆技术,但它存在处理数据和检测方面的局限性。例如,动态获取每个执行路径的行为特征需要大量的资源和时间。动态分析仅考虑单一路径的行为特征^[11],因此存在代码覆盖率不足的问题。另外,许多现代恶意软件采用反分析技术,在模拟环境中能够成功隐藏其恶意行为^[12]。

静态分析能够在不执行程序的情况下,通过直接分析代码和程序的内部结构来检查潜在的恶意行为。这种分析方式不仅大幅度节省了处理数据的时间,而且能够实现对程序代码的全覆盖检查。与此同时,静态分析不受恶意软件使用的反分析技术或特定执行环境的影响,使其在识别复杂和隐蔽的恶意软件方面显示出独特的优势。具

体而言,静态分析通过深入挖掘恶意软件的内部特征,如字节码^[13]、汇编指令^[14]和文件特征^[15]等,能够提供一种有效的检测手段。例如, Du 等^[15]使用递归特征消除(RFE, recursive feature elimination)和主成分分析(PCA, principal component analysis)提取静态特征,提出了一种基于最邻近(KNN, k-nearest neighbor)算法和密度的机器学习方法来预测勒索软件攻击,实验结果表明,该方法能显著提高恶意软件检测的准确性。

尽管静态分析可以在不污染系统的情况下实现全路径覆盖,但低检测率和高误报率是一个棘手的问题。为了解决该问题,研究者开始考虑使用多个特征进行恶意软件检测^[2,16-17]。例如, Kim 等^[16]专注于 Android 移动恶意软件的检测。他们采用了多特征融合的深度学习方法,利用了多种静态特征,如字符串、操作码(Opcode)、API 调用、权限、组件和环境等。这种深入的特征提取方法使他们能够对大量样本进行精确的恶意软件检测。Han 等^[2]提出了名为 MallInsight 的程序分析模型,采用了动静结合的思想,旨在实现对恶意软件的有效检测,不仅关注恶意软件的基本结构,还深入探讨了程序与操作系统之间的交互,以及对文件、注册表和网络的操作。然而,当前恶意软件检测领域多特征融合方法仍局限于同一模态特征,无法多角度地从不同模态表示样本信息。另外,现有特征融合方法存在局限性,即无法同时学习各个特征内部和特征之间的联系,这将导致跨模态特征融合时特征内部信息丢失的问题。

为了解决上述问题,本文提出了一个多模态特征融合的深度学习方法——基于注意力与门控机制的多特征融合恶意软件检测方法(MFAGM, multi-feature fusion malware detection method based on

attention and gating mechanism)。首先, 针对当前恶意软件检测方法特征模态选取单一的问题, 该方法使用 API、操作码和图像信息作为输入特征, 从多角度表示恶意软件样本信息。然后, 针对现有特征融合方法存在特征内部信息丢失问题, 本文设计了特征融合模块——具有跳跃链接的自注意增强门控机制单元 (SA-JGmu, self-attention enhanced gate mechanism unit with jump connection) 进行特征融合, 该模块综合考虑了特征内部的相关性以及特征之间的相互影响。本文实验证明, 所提方法在恶意软件分类任务中取得了更高的准确率和 $F1$ 分数。本文的贡献点如下。

1) 将数据集中的 .asm 文件通过静态分析获取其操作码统计特征和 API 序列特征, 并将 .bytes 文件处理成灰度图像表示文件信息, 用两种类型的 3 种特征表示该恶意软件。

2) 提出了 MFAGM。该方法旨在有效整合多种模态特征, 特别强调考虑了特征内部的关联性 & 特征间的相互作用。所提特征融合模块 SA-JGmu 采用自注意力机制来挖掘特征内部的关系, 并通过门控机制探索不同特征间的交互效果。此外, 该模块运用跳跃连接策略, 以保留部分原始特征, 进而增强了模型对整体特征的代表能力。

3) 使用微软恶意软件分类挑战 (MMCC, Microsoft malware classification challenge) 数据集 BIG2015 进行实验, 结果显示 MFAGM 达到了更高的准确率和 $F1$ 分数, 证明了所提方法的有效性。

1 相关工作

通常, 恶意软件检测领域的研究包括两种基本方法: 静态分析和动态分析。本文主要集中于静态分析方法, 并阐述该领域的相关研究工作。

1.1 基于单一特征的恶意软件检测方法

静态分析方法常常对表示恶意软件的 .bytes 文件和 .asm 文件进行处理并提取特征。首先, .bytes 文件可以重新组织为灰度图像, 其中每个字节都被解释为图像中的一个像素, 值范围为 0 到 255 (0: 黑色, 255: 白色)。例如, Gibert 等^[18]使用灰度图像作为输入, 利用卷积神经网络架构提取特征, 从而实现了对恶意软件的检测。Narayanan 等^[19]通过 PCA 的方法对灰度图像进行

特征提取, 并运用机器学习模型进行检测。另外, Wojnowicz 等^[20]提出了一种方法来量化文件结构熵的变化以判断该文件的可疑程度。可执行文件结构熵的计算过程即将文件分成固定长度的不重叠的块, 随后对每个块测量它的熵。因此, 每个文件都可以表示为熵的时间序列, 作为恶意软件检测的特征之一。例如, Gibert 等^[21]提出了一种基于卷积神经网络的系统, 以结构熵作为模型输入, 将恶意软件分为不同的家族。此外, 许多研究者^[22-25]直接使用字节序列作为特征进行恶意软件检测。

.asm 文件包含许多有价值的信息, 可以作为恶意软件分类的依据。例如, API 函数的调用提供了有关操作系统提供的服务和资源的信息, 而操作码序列可以表示为恶意软件的底层行为特征。Ahmadi 等^[26]以 API 和操作码序列作为输入, 使用极端梯度增强 (XGBoost, extreme gradient boosting) 算法来进行恶意软件的检测工作。McLaughlin 等^[27]和 Gibert 等^[14]仅使用了操作码序列作为输入, 利用卷积神经网络 (CNN, convolutional neural network) 对恶意软件进行检测。

1.2 基于多特征的恶意软件检测方法

对恶意软件分类时, 可以提取多种类型数据的特征共同作为恶意软件检测的依据。如果仅将汇编语言指令和原始二进制序列或其压缩表示作为输入, 则很多用于表征恶意软件的有用信息将被忽略。因此, 与多模态方法相比, 这些只使用单一信息源作为输入的单模态深度学习方法往往表现不佳。这是因为不同类型的特征或模态能够提供互补的信息, 并揭示出单独使用单个模态时无法发现的模式。Gibert 等^[17]采用字节特征、操作码特征和 API 特征的单一模态特征作为输入, 应用深度学习方法对恶意软件进行检测。Mays 等^[28]以图像和操作码作为输入, 运用集成学习的方法进行恶意软件检测。尽管这些采用多特征的方法都达到了不错的分类效果, 然而, 它们普遍存在一个问题, 即无法同时学习各个特征内部和特征之间的联系, 难以将各个特征进行有效融合。

综上所述, 本文提出了 MFAGM。该方法使用 SA-JGmu 融合模块对两种类型的 3 种特征进行有效融合, 并对恶意软件样本进行分类, 提高了分类的准确率和 $F1$ 分数。

2 本文方法

2.1 整体框架

本文方法对 BIG2015 数据集处理如下。首先,对数据集进行预处理。从 .asm 文件中提取 API 函数序列和操作码统计序列,并将 .bytes 文件转化为灰度图像。其次,将处理好的 3 种数据输入

MFAGM 中。利用 3 个子模型分别提取 API 序列特征、操作码统计特征和灰度图像特征,这 3 个子模型分别是 API 网络、Opcode 网络和 Image 网络。接着,将提取的 3 种特征输入特征融合模块 SA-JGmu 中,以实现有效的特征融合。最后,将融合后的特征输入分类模块,以便进行恶意软件

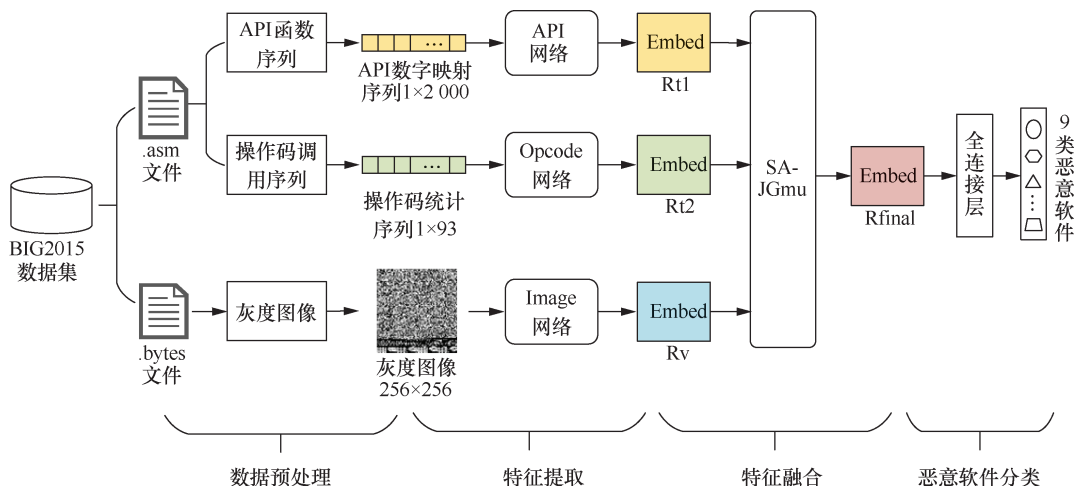


图 1 所提方法总体架构
Figure 1 Overall architecture of the proposed method

此外,为了直观地展示方法的运作机制,本文后续将提供相关的数学公式描述。

2.2 数据预处理

本节主要介绍如何处理 BIG2015 数据集^[29],以获取子模型能够识别和处理的数据格式。

2.2.1 API 数字映射序列

API 函数是与操作系统提供的服务紧密相关的。它们为软件提供了访问操作系统管理的系统资源的唯一途径,涵盖网络、安全、系统服务和文件管理等关键操作。因此,API 函数序列为本文方法提供了描述特定软件意图或行为的描述性信息。

为了能够处理这些序列,本文方法进行了一系列的数据处理,步骤如下:首先,采用 Gibert 等^[30]提供的方法 PE Parse 对 .asm 文件进行处理,获得 API 函数序列;其次,为所有的 API 函数名称创建了字典映射,其中每个 API 函数名都被赋予了唯一的数字标识符;最后,将 API 函数序列从文本形式转化为数字序列。

在处理的过程中,为确定 API 函数序列长度,

本文方法对数据集进行了详尽的统计分析。序列长度范围统计如图 2 所示,约 66.25% 的序列长度不超过 500, 88.38% 的序列长度不超过 1 000,而高达 97.36% 的序列长度则不超过 2 000。为了标准化输入,本文将所有 API 序列长度定为 2 000。对于长度小于 2 000 的序列,本文采用零填充的方法进行补齐;而对于长度超过 2 000 的序列,本文进行了截断处理,最终得到了固定长度为 2 000 的 API 数字映射序列。

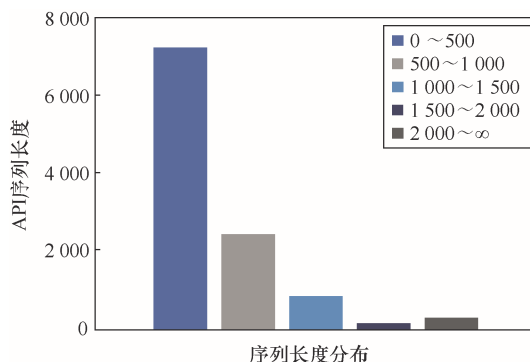


图 2 序列长度范围统计
Figure 2 Statistical of sequence length range

2.2.2 操作码统计序列

操作码作为机器语言的基本指令，为恶意软件的行为提供了直观的描述。在恶意软件的执行过程中，其操作码的模式和频率可能在不同类别的恶意软件中存在显著差异。因此，利用操作码的频率作为特征进行恶意软件分类具有很大的潜力。

首先，本文采用了 Gibert 等^[30]提供的方法 PE Parse 处理 .asm 文件。具体而言，该方法根据恶意软件中操作码的出现频率提取了频率最高的 93 个操作码^[31]，这些操作码的总出现频率占据了 99.8% 以上^[32]，随后与 .asm 文件内容进行匹配搜

索，以提取出操作码调用序列。然后，本文对调用序列进行频率统计，记录每个操作码的出现频率，最终生成一个长度为 93 的数字频率序列，作为其子模型的输入。值得注意的是，在进行操作码频率统计之前，本文固定了操作码统计序列的前后顺序，以得到一个按照固定顺序排列的操作码频率序列。操作码序列提取流程如图 3 所示。这种方法的优势在于：它不仅捕获了恶意软件使用的特定操作码，还捕获了这些操作码的相对重要性。例如，某些恶意操作可能会频繁地使用特定的操作码，而这种情况在某些软件中不常见。

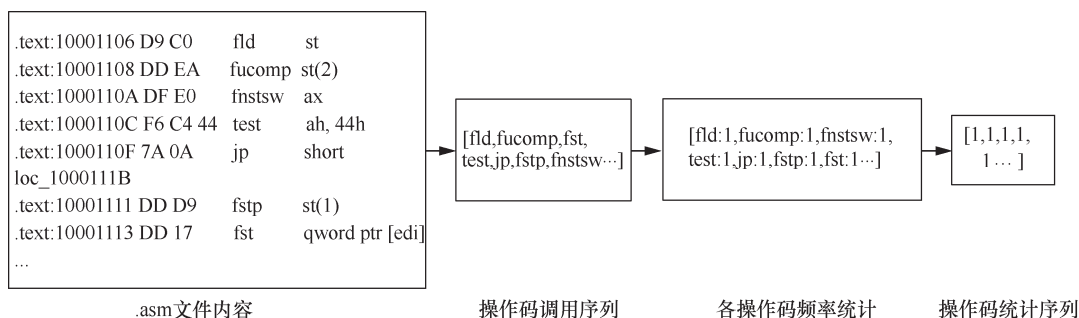


图3 操作码序列提取流程

Figure 3 Opcode sequence extraction process

2.2.3 灰度图像

灰度图像作为静态分析的结果，提供了一种对 .bytes 文件可视化的特征表示。它将 .bytes 文件的每个字节解释为像素，并将其映射到灰度值。灰度图像捕捉了 .bytes 文件的结构和布局信息，其中包含不同类别的恶意软件之间的差异，以及同类的软件样本灰度图像的相似信息。

为了获取灰度图像，本文采取的数据处理步骤如下：首先，去除 .bytes 文件第一列的偏移量信息，仅保留真正的内容部分；接着，将每两位十六进制数字组合成一组，并将其转化成对应的十进制数字；随后，将这些数字转置为二维数组，其中每个十进制数字代表一个灰度像素，像素值的范围为 0（黑色）到 255（白色）。由于 .bytes 文件的大小可变（通常约为几十 kB），它们不能完全适合固定大小的图像。因此，本文只从每个

样本中提取 n^2 个初始字节，之后映射到正方形 $n \times n$ 的图像。例如，对于 $n=64$ ，则提取前 4 096 个字节。请注意，本文从文件的开头提取这些字节，而不是从任意位置提取这些字节，是因为该区域包含可执行文件的特征，如程序头和 PE 头。生成灰度图像流程如图 4 所示。

由于各类 .bytes 文件的大小范围存在较大差异，因此在将其转化为灰度图像时，某些文件的大小可能不足以满足转化所需的字节量。由图 5 所示的图像尺寸分布的统计结果可知，100% 的 .bytes 文件可以转化成 128×128 的灰度图像，99% 可以转化成 256×256 的灰度图像，86.5% 可以转化成 512×512 的灰度图像。为了使灰度图像保留更多的信息以及考虑到实验效果，经过验证，本文最终选择将软件样本转化成 256×256 的灰度图像，使用零补法，将大小不够的用零补齐，超过的直接截断。

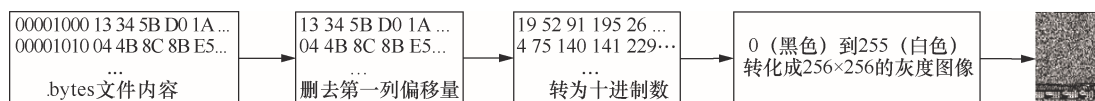


图4 生成灰度图像流程

Figure 4 Process for generating grayscale images

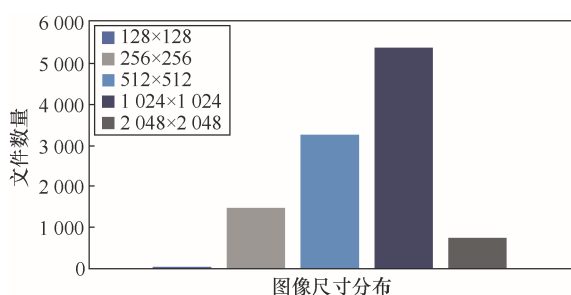


图 5 图像尺寸分布
Figure 5 Image size distribution

2.3 特征提取

通过数据预处理, 本文成功将原始数据集转化为 3 种不同格式的表示: 长度为 2 000 的 API 数字映射序列、长度为 93 的操作码统计序列, 以及 256×256 的灰度图像。随后, 这些数据被分别输入相应的子模型中, 以进行特征提取。

2.3.1 API 特征提取

在恶意软件分析领域, API 函数序列的深度特征提取显得尤为关键。本文基于 TextCNN^[33] 的思想, 设计出一个卷积神经网络 (API 网络) 来提取 API 序列特征。

首先, API 函数序列通过嵌入层被映射到一个连续的向量空间。具体来说, 每个数字被嵌入为一个 4 维向量, 从而将整个序列转化为一个 4×2 000 的矩阵作为模型的输入。其次, 该子模型使用 3 个不同大小的卷积核, 每个过滤器的大小是 $h \times k$, 其中 $h \in \{3, 4, 5\}$, $k=4$, 在卷积层中捕获 API 函数序列的多尺度局部模式。每个卷积层的输出经过最大池化操作, 以提取最显著的特征

并减少数据维度。随后, 3 个卷积层的输出经过拼接, 并通过全连接层进一步提取高级特征。为防止过拟合, 在全连接层后加入了 Dropout 层。最终输出一个 256 维的特征向量, 记为 $R_{t1} \in R^{256 \times 1}$ 。通过这一系列操作, API 网络成功地从 API 函数序列中提取了深度特征, 为后续的恶意软件分类提供了关键信息。API 网络架构如图 6(a) 所示。

2.3.2 操作码特征提取

为了精确抽取操作码统计特征中的关键信息, 本文构建了一个专门的神经网络子模型, 命名为 Opcode 网络, 具体架构如图 6(b) 展示。

输入特征: 模型接收一个长度为 93 的数字频率序列作为输入。这个序列代表操作码的统计特征, 其中每个数字代表特定操作码的出现频率。

隐藏层: 该层是模型的核心部分, 它将输入特征转化为一个中间的隐藏表示。本文使用了线性 (Linear) 变换, 其参数由输入特征的数量和隐藏层神经元的数量定义。

激活函数: 为了引入非线性并增强模型的表示能力, 该子模型在隐藏层后使用了 ReLU 激活函数。

输出 Dropout 层: 该子模型在隐藏表示后添加了 Dropout 层, 增强了模型的鲁棒性。

最终, Opcode 模型输出一个 256 维度的特征向量, 记为 $R_{t2} \in R^{256 \times 1}$, 为后续的特征融合做准备。该 256 维度的向量捕获了操作码统计特征的关键信息, 为恶意软件检测任务提供了有力支持。

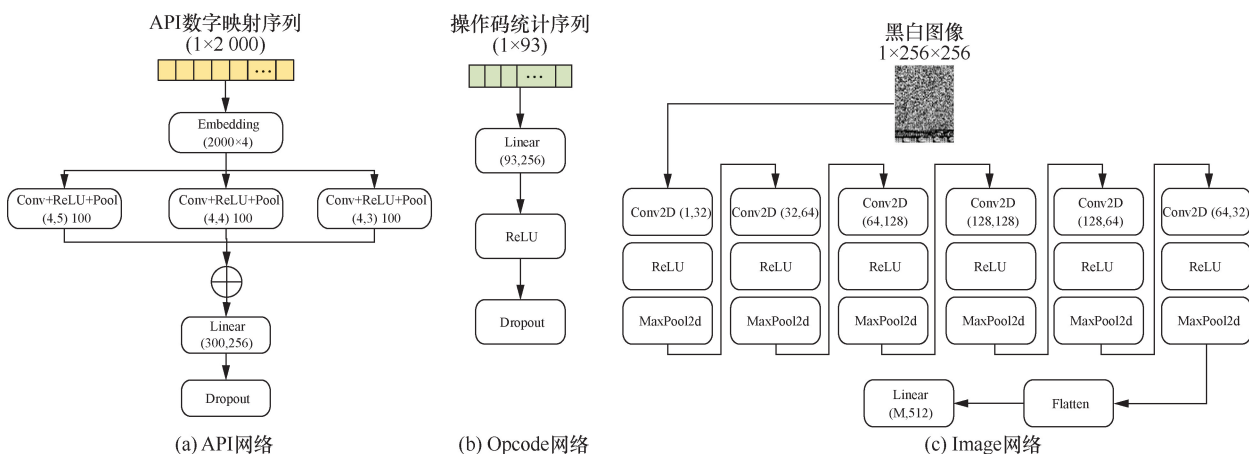


图 6 各子模型架构
Figure 6 Architecture of each sub model

2.3.3 灰度图像特征提取

在恶意软件检测任务中，本文还考虑了另一种特征：256×256 的灰度图像。这些图像为本文提供了恶意软件的视觉表示，反映了其内部结构和模式。为了从这些图像中提取有意义的特征，本文设计了一个名为 Image 网络的卷积神经网络模型，Image 网络架构如图 6(c)所示。

特征提取：模型首先通过一系列的卷积层进行特征提取，每层网络的卷积核大小为 3×3，步长为 1。每个卷积层后都跟随一个 ReLU 激活函数，以增加模型的非线性表示能力。为了减少特征的空间维度并捕获更大范围的上下文信息，该子模型在每个卷积层后引入了最大池化层，池化核大小为 2×2，步长为 1。

全连接层：经过特征提取后，模型使用全连接层将卷积特征转化为 512 维度的特征向量，记为 $R_v \in R^{512 \times 1}$ 。这个向量捕获了图像的关键信息，并为后续的特征融合做准备。

2.4 特征融合

在恶意软件检测中，API 函数序列、操作码统计和灰度图像特征都是至关重要的特征。它们各自揭示了软件的行为、结构和视觉模式。尤其是 API 函数序列，由于其顺序性，可能存在长距离的依赖关系。为了深入挖掘这些特征的潜力并实现它们的优化表示，本文设计了一个特征融合模块 SA-JGmu。

2.4.1 注意力机制

本文采用自注意力 (SA, self-attention) 机制^[34]对每种特征进行处理。SA 不仅有效地捕获了 API 函数序列中的长距离依赖，而且强化了各个特征之间的内部联系和依赖性。

本文方法使用多头自注意力 (MSA, multi-head self-attention) 模块，SA 架构如图 7 所示，它由两个子层组成。第一个子层是多头注意力机制，第二个子层是一个简单的全连接前馈神经网络 (FFN, feed-forward neural network)。在每个子层之后都会执行残差连接和归一化。每个层的输出都是 $\text{LayerNorm}(\mathbf{x} + \text{SubLayer}(\mathbf{x}))$ ，其中 $\text{SubLayer}(\mathbf{x})$ 是指由子层实现的函数。

注意力函数可以描述为将查询向量和一组键值对向量映射到输出向量。计算输入以获得 $d \times 1$

维的查询向量、键和值，将它们分别加载到矩阵 Q 、 K 和 V 中。通过 Q 和 K 之间的点积相似度，确定分配给 V 的权重，注意力模块的输出如式(1)所示。

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \quad (1)$$

多头注意力机制拥有 m 个头和 m 个平行的自注意力模块。对于第 i 个头， Q 、 K 和 V 输入后的转化如式(2)所示。

$$Q_i = QW_i^Q, K_i = KW_i^K, V_i = VW_i^V \quad (2)$$

其中， W_i^Q 、 W_i^K 、 $W_i^V \in R^{l \times d_h}$ 是第 i 个头的投影矩阵， $d_h = d / m$ 是每个头的输出特征的维度。

多头注意力机制计算如式(3)~式(4)所示。

$$\text{MultiHead}(Q, K, V) = \text{Concat}(h_1, \dots, h_m)W^O \quad (3)$$

$$h_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (4)$$

其中， $W^O \in R^{md_h \times 1}$ 。

全连接前馈网络使用两个线性变换，中间带有 ReLU 激活函数，如式(5)所示。

$$\text{FFN}(\mathbf{x}) = \max(0, \mathbf{x}W_1 + b_1)W_2 + b_2 \quad (5)$$

其中，输出是 $d \times 1$ 维。

本文用式(6)描述上述整个过程。

$$\mathbf{x}' = \text{MSA}(\mathbf{x}) \quad (6)$$

其中， $\mathbf{x}'$ 是 MSA 的输出， $\mathbf{x}' \in R^{d \times 1}$ 。

通过上文的描述，各个子模型的输出分别是灰度图像特征、API 序列特征和操作码统计特征，表示为 R_v 、 R_{t1} 和 R_{t2} 。将它们分别输入 SA 中，如式(7)所示。

$$R'_i = \text{MSA}(R_i) \quad (7)$$

其中， $i \in \{v, t1, t2\}$ ， R'_v 、 R'_{t1} 和 R'_{t2} 分别为经过 SA 处理后的灰度图像特征、API 序列特征和操作码统计特征。

2.4.2 门控机制&跳跃连接

在恶意软件检测的实验中，3 种特征（灰度图像特征、API 序列特征、操作码统计特征）在各个子模型中显示出不同的分类准确率，这表明 3 个子模型的表示能力有所不同，所以这些特征被融合以进行最终的分类决策时，应该赋予不同的权重。

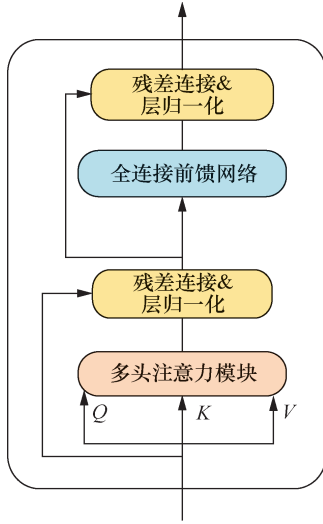


图 7 SA 架构
Figure 7 SA Architecture

本文方法利用了门控多模态单元 (GMU, gated multimodal unit) [35] 来融合经过 SA 处理后的 3 种特征。GMU 的核心思想是为每种特征动态地分配权重, 这些权重是基于输入数据学习得到的。通过这种方式, 本文方法可以根据每个样本的特性自适应地调整特征的权重, 从而实现更高的分类准确率。

将经过 SA 处理后的特征 R'_v 、 R'_{t1} 和 R'_{t2} 输入 GMU 中, 将特征与权重系数相乘并激活, 得到各个特征的激活表示, 如式(8)所示。

$$h_i = \tanh(W_i \cdot R'_i) \quad (8)$$

其中, $i \in \{v, t1, t2\}$, $W_i = \text{Linear}(R_i \text{dim}, \text{hdim})$, $\text{hdim} = 256$, 表示各个模态特征的权重。 h_v 、 h_{t1} 和 h_{t2} 分别表示为灰度图像特征、API 序列特征和操作码统计特征激活后的特征表示。

随后, 计算各个特征的门值, 这是门控机制的重点, 如式(9)~式(11)所示。

$$\text{CI} = \text{Concat}((R'_v, R'_{t1}, R'_{t2}), \text{dim} = 1) \quad (9)$$

$$z_i = \sigma(W_i \cdot \text{CI}) \quad (10)$$

$$h_{\text{GMU}} = z_v \cdot h_v + z_{t1} \cdot h_{t1} + z_{t2} \cdot h_{t2} \quad (11)$$

其中, CI 是 3 个特征的拼接, $i \in \{v, t1, t2\}$, $W_i = \text{Linear}(R_i \text{dim} + R_2 \text{dim} + R_3 \text{dim}, \text{hdim})$, $\text{hdim} = 256$, 为各个特征的门控权重。 z_v 、 z_{t1} 和 z_{t2} 分别是灰度图像特征、API 序列特征和操作码统计特征的门值, h_{GMU} 为 GMU 的输出。

最后, 为了进一步增强模型的表示能力, 本文将 3 个子模型提取出的原始特征进行拼接, 并经过 SA 处理进一步提取出原始特征信息, 随后与 GMU 的输出进行权重拼接, 本文称之为跳跃连接 (JC, jump connection)。经过实验验证, 本文将 $\alpha=0.05$ 的权重分配给原始特征, $1-\alpha=0.95$ 的权重分配给 GMU 的输出 h_{GMU} 。这样的设计确保了模型在从 GMU 中获取融合信息的同时, 能直接利用原始特征, 从而实现更为全面和细致的特征表示, 如式(12)~式(14)所示。

$$\text{shortcut} = \text{Linear}(\text{CI}, \text{hdim}) \quad (12)$$

$$\text{shortcut}' = \text{MSA}(\text{shortcut}) \quad (13)$$

$$R_{\text{final}} = \alpha \cdot \text{shortcut}' + (1-\alpha) \cdot (z_v \cdot h_v + z_{t1} \cdot h_{t1} + z_{t2} \cdot h_{t2}) \quad (14)$$

其中, shortcut 表示经过线性变换后的 CI , 这一变换通过全连接层实现, 将输入特征映射到 $\text{hdim} = 256$ 维空间。 $\text{shortcut}'$ 为跳跃连接得到的特征, R_{final} 为经过 SA-JGmu 融合模块的输出。

SA-JGmu 架构如图 8 所示。

3 实验

3.1 数据集

本文实验使用 Kaggle 提供的 MMCC 数据集 BIG2015^[29], 该数据集包含代表 9 种不同恶意软件系列的示例, 其中每个示例都关联了两个文件: ① 恶意软件二进制文件, 即 .bytes 文件; ② 其相应的汇编语言源代码文件, 即 .asm 文件, 它是使用 IDA Pro 反汇编程序生成的。该数据集的恶意软件类别及数量如表 1 所示。

3.2 实验环境

本文所提框架是使用 Python 3.8.10 和 Pytorch 1.11.0 开发的, 因为它具有强大的 GPU 加速张量计算能力和自动推导机制。它被部署在一台带有 12 vCPU Intel Xeon Silver 4214R CPU @ 2.40 GHz, 90.0 GB RAM 和 RTX 3080 Ti(12 GB)的机器上。

3.3 实验指标

在分类任务中, 常用的评价指标有准确率 (Accuracy)、召回率 (Recall)、精确率 (Precision) 和 F1 分数 (F1-score)。本文计算了在测试集上

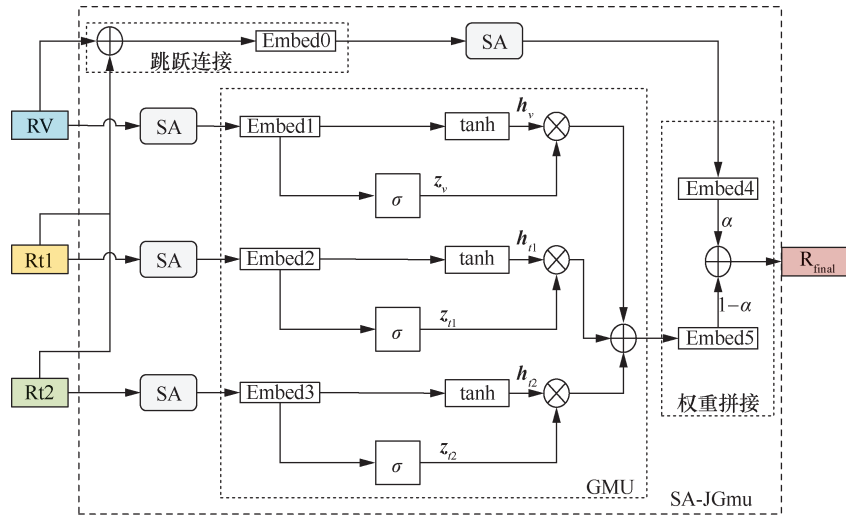


图 8 SA-JGmu 架构
Figure 8 SA-JGmu architecture

的真阳性 (TP)、假阳性 (FP)、真阴性 (TN) 和假阴性 (FN) 的数量。这些评价指标的计算方法如式(15)~式(18)所示。

表 1 恶意软件类别及数量
Table 1 Types and quantity of malware

家族	数量/个	类别
Ramnit	1 541	蠕虫
Lollipop	2 478	广告软件
Kelihos_ver3	2 942	后门
Vundo	475	木马
Simda	42	后门
Tracur	751	木马下载器
Kelihos_ver1	398	后门
Obfuscator.ACY	1 228	混淆的恶意软件
Gatak	1 013	后门

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \quad (15)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (16)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (17)$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (18)$$

3.4 实验结果

3.4.1 子模型与完整模型实验结果对比

为了全面评估 MFAGM 的有效性, 本文对完

整模型进行了性能评估。通过多模态特征的融合, 模型能够更准确地进行恶意软件分类。为了验证这一点, 本文在预处理后的数据集上进行了实验, 并按照 8:2 的比例将数据集划分了训练集和测试集。子模型与完整模型实验结果对比如表 2 所示。

表 2 子模型与完整模型实验结果对比
Table 2 Comparison of experimental results between sub models and complete models

模型	准确率	F1 分数	精确率	召回率
API 网络	97.29%	85.71%	85.14%	86.38%
Opcode 网络	98.02%	97.39%	98.08%	96.81%
Image 网络	96.50%	93.26%	93.86%	92.76%
MFAGM 的完整模型	99.17%	99.03%	99.03%	99.02%

从实验结果中可以明显看出, 单独使用 API 序列特征 (API 网络)、操作码统计特征 (Opcode 网络) 或灰度图像特征 (Image 网络) 的子模型在所有指标上均不如本文方法的完整融合模型。而本文方法的各个指标都在 99% 以上, 这意味着本文方法不仅能够准确地分类恶意软件, 而且在平衡误报和漏报方面做得非常好。实验结果进一步证明了多模态特征融合策略的有效性。通过结合 API、操作码和图像特征, MFAGM 能够充分利用这些特征的互补性, 从而实现更高的分类性能。

3.4.2 对比实验

为了进一步评估本文方法的性能, 将 MFAGM 与文献中的方法进行了比较, 这些方法

同样在恶意软件分类挑战提供的数据集 BIG2015 上评估了它们的模型。对比实验结果如表 3 所示。

如图 9 所示, MFAGM 在恶意软件检测任务中展现出卓越的性能。与选定的其他方法相比, MFAGM 在准确率和 $F1$ 分数上均高于其他方法。这种性能的提升主要归功于模型在整合和学习操作码、API 函数序列和图像数据方面的高效性。这 3 种数据源均独立且富含信息, 它们的结合为模型提供了强大的判别力。

特别地, MFAGM 采用自注意力机制和门控机制进行特征融合, 并通过跳跃连接策略有效地保留了部分原始特征, 从而在特征内部及其相互

作用上实现了精细的学习过程。相比传统的单一特征提取方法^[14,18-19,21-27]、多特征的机器学习方法^[28], 以及仅使用基础融合技术多特征融合的深度学习方法^[17], MFAGM 更能全面捕捉恶意软件的潜在模式, 并从多维度来理解恶意软件行为, 在实验中获得了更高的准确率和 $F1$ 分数。这进一步验证了在恶意软件分类任务中, 多模态特征融合及 SA-JGmu 融合模块的有效性。

3.4.3 消融实验

为了证明本文方法中每个组件的有效性, 本文将每个组件从整个模型中单独删除以进行比较。All 表示包含所有组件的模型, 包括 API 序

方法	特征类型	算法	准确率	$F1$ 分数
灰度图像				
Gibert ^[18]	128×128 Grayscale Image	CNN	97.50%	94.00%
Ahmadi ^[26]	Haralick features	XGBoost	96.90%	92.82%
Ahmadi ^[26]	Local Binary Pattern features	XGBoost	97.24%	95.30%
Narayanan ^[19]	PCA features	1-NN	96.60%	91.02%
熵				
Gibert ^[21]	Structural Entropy	CNN	98.28%	96.36%
Ahmadi ^[26]	Entropy Statistical Measures	XGBoost	99.00%	97.66%
操作码序列				
McLaughlin ^[27]	Opcode sequence	CNN	99.03%	97.43%
Gibert ^[14]	Opcode sequence	Hierarchical CNN	99.13%	98.30%
字节序列				
Ahmadi ^[26]	1-Gram	XGBoost	98.50%	96.78%
Rafi ^[22]	Bytes sequence	MalConv	96.41%	88.94%
Le ^[23]	Scaled bytes sequence	CNN	96.47%	93.41%
Le ^[23]	Scaled bytes sequence	CNN+Unidirectional LSTM	98.00%	95.77%
Le ^[23]	Scaled bytes sequence	CNN+Bidirectional LSTM	98.14%	96.62%
Gibert ^[24]	Bytes sequence	Denoising Autoencoder + Dilated Residual Network	98.61%	97.19%
Yousefi-Azar ^[25]	1-Gram	Autoencoder + XGBoost	93.09%	86.64%
API				
Ahmadi ^[26]	API feature vector (796)	XGBoost	98.68%	96.38%
多特征				
Mays ^[28]	IMG and Opcode N-Grams	Ensemble Learning (CNN and NN)	97.24%	96.18%
HYDRA ^[17]	APIs, Bytes sequence, Opcode sequence	Multi-feature fusion Network	98.71%	96.95%
MFAGM	API sequence features, opcode statistical features, 256×256 grayscale images	Multimodal feature fusion with SA-JGmu Network	99.17%	99.03%

图 9 实验结果对比

Figure 9 Comparison of experimental results

列特征(A)、操作码统计特征(O)、灰度图像特征(G)、自注意力模块(SA)、门控模块(GMU)、跳跃连接(JC)和本文的融合模块(SA-JGmu)。本文使用-A、-O、-G、-SA、-GMU和-JC分别表示从整个模型中去掉相应的模块，-SA-JGmu表示去掉本文的特征融合模块。实验结果进一步证明本文特征融合模块的有效性。消融实验的结果如表3所示。

表3 消融实验的结果
Table 3 Results of ablation experiment

模型	准确率	F1 分数	精确率	召回率
-A	98.94%	98.26%	97.75%	98.82%
-O	97.75%	93.99%	92.69%	96.52%
-I	98.85%	97.17%	95.89%	98.87%
-SA	98.71%	96.72%	95.63%	98.45%
-JGmu	98.99%	97.62%	97.64%	97.60%
-JC	99.03%	98.30%	97.69%	98.99%
-SA-JGmu	98.34%	98.08%	98.10%	98.11%
All	99.17%	99.03%	99.03%	99.02%

通过消融实验可以看出，本文方法中的所有组件都有助于提升模型的性能。首先，本文关注多个特征，从不同子模型的原始数据中提取了3个特征，且在数据集上的实验证明了多个特征融合的有效性。在去除任何一个特征、仅使用另外两个特征进行融合的情况下，所有的评价指标值都会相应降低，证明本文方法的有效性。具体来说，操作码统计特征和灰度图像特征的贡献大于API序列特征，这意味着.bytes文件表示的灰度图像包含了重要的信息。

此外，通过-SA和-JGmu实验结果可知，当不使用注意力或门控机制时，所有实验指标都有所下降，这能够证明注意力和门控机制的有效性。通过-JC实验，发现跳跃连接将原始特征保留在最后的特征表示中，使分类效果有所提升，这证明了跳跃连接的有效性，即保留一部分原始特征能够有效提升恶意软件分类效果。通过-SA-JGmu实验结果表明，SA-JGmu相较于传统拼接方法，在特征融合方面表现出显著的优势。具体地，当去除SA-JGmu模块，仅使用简单的拼接方法进行特征融合时，模型的性能指标出现了不同程度的下降。这一结果说明，SA-JGmu不仅在特征尺度

上实现了更加精确的匹配，而且在特征交互和关系学习方面展现了更高的效率。总体而言，消融实验证明了SA-JGmu在多模态深度学习架构中的核心作用，它是本文方法获得优异性能的关键因素之一。

4 结束语

本文提出一种基于注意力和门控机制的多特征融合恶意软件检测方法，以解决恶意软件检测中的多特征提取和融合问题。与传统方法不同，本文从.asm文件提取了可以代表软件行为的API序列特征和操作码统计特征，并从.bytes文件中提取灰度图像特征，最终进行多模态特征融合。为了更好地融合这3种特征，本文设计了一个特征融合模块SA-JGmu，该模块采用自注意力机制捕获特征内部的依赖关系，并通过门控机制强化不同特征之间的交互。此外，在进行特征融合时，该方法通过跳跃连接，直接将原始特征与GMU融合后的特征进行权重拼接，使最终得到的特征保留了一部分原始特征信息，进一步加强了模型的表示能力。实验结果表明，MFAGM在准确率、F1分数、精确率和召回率4个关键指标上具有明显优势，证明了该方法在恶意软件检测领域的有效性和可靠性。

通过多模态特征融合与注意力和门控机制的应用，MFAGM可以更准确地检测恶意软件，提高计算机的安全性。这种方法不仅适用于恶意软件检测，还可以应用于其他领域的多模态特征融合问题。在未来的工作中，将继续对特征选择进行优化，并探索动态和静态特征的结合，以实现恶意软件样本的更全面表征。同时，将进一步改进多模态特征融合模块，以提升恶意软件检测的效果。

参考文献：

- [1] BlackBerry. Global threat intelligence report [R]. 2023-11.
- [2] HAN W J, XUE J F, WANG Y, et al. MalDAE: detecting and explaining malware based on correlation and fusion of static and dynamic characteristics[J]. Computers & Security, 2019, 83: 208-233.
- [3] KORCZYNSKI D, YIN H. Capturing malware propagations with code injections and code-reuse attacks[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 1691-1708.

- [4] RIZVI S K J, ASLAM W, SHAHZAD M, et al. PROUD-MAL: static analysis-based progressive framework for deep unsupervised malware classification of windows portable executable[J]. *Complex & Intelligent Systems*, 2022, (2022): 1-13.
- [5] JOHNSON S, GOWTHAM R, NAIR A R. Ensemble model ransomware classification: a static analysis-based approach[M]//*Inventive Computation and Information Technologies: Proceedings of ICICIT 2021*. 2021: 153-167.
- [6] LOI N, BORILE C, UCCI D. Towards an automated pipeline for detecting and classifying malware through machine learning[J]. *ArXiv Preprint arXiv:2106.05625*, 2021.
- [7] BARBI S, BARBIERI F, MARINELLI S, et al. Phase change material-sand mixtures for distributed latent heat thermal energy storage: Interaction and performance analysis[J]. *Renewable Energy*, 2021, 169: 1066-1076.
- [8] CHANAJITT R, PFAHRINGER B, GOMES H M, et al. Multiclass malware classification using either static opcodes or dynamic API calls[C]//*Proceedings of Australasian Joint Conference on Artificial Intelligence*. 2022: 427-441.
- [9] JING C, WU Y, CUI C Y. Ensemble dynamic behavior detection method for adversarial malware[J]. *Future Generation Computer Systems*, 2022, 130: 193-206.
- [10] 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.
- [11] AFIANIAN A, NIKSEFAT S, SADEGHIYAN B, et al. Malware dynamic analysis evasion techniques: a survey[J]. *ACM Computing Surveys (CSUR)*, 2019, 52(6): 1-28.
- [12] LEBBIE M, PRABHU S R, AGRAWAL A K. Comparative analysis of dynamic malware analysis tools[C]//*Proceedings of the International Conference on Paradigms of Communication, Computing and Data Sciences: PCCDS 2021*. 2021: 359-368.
- [13] GAO X W, HU C Z, SHAN C, et al. Malware classification for the cloud via semi-supervised transfer learning[J]. *Journal of Information Security and Applications*, 2020, 55: 102661.
- [14] GIBERT D, MATEU C, PLANES J. A hierarchical convolutional neural network for malware classification[C]//*Proceedings of 2019 International Joint Conference on Neural Networks (IJCNN)*. 2019: 1-8.
- [15] DU J, RAZA S H, AHMAD M, et al. Digital forensics as advanced ransomware pre-attack detection algorithm for endpoint data protection[J]. *Security and Communication Networks*, 2022, (2022): 1-16.
- [16] KIM T G, KANG B J, BHO M, et al. A multimodal deep learning method for android malware detection using various features[J]. *IEEE Transactions on Information Forensics and Security*, 2018, 14(3): 773-788.
- [17] GIBERT D, MATEU C, PLANES J. HYDRA: a multimodal deep learning framework for malware classification[J]. *Computers & Security*, 2020, (95): 101873.
- [18] 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): 15-28.
- [19] NARAYANAN B N, DJANEYE-BOUNDJOU O, KEBEDE T M. Performance analysis of machine learning and pattern recognition algorithms for malware classification[C]//*Proceedings of 2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS)*. 2016: 338-342.
- [20] WOJNOWICZ M, CHISHOLM G, WOLFF M, et al. Wavelet decomposition of software entropy reveals symptoms of malicious code[J]. *Journal of Innovation in Digital Ecosystems*, 2016, 3(2): 130-140.
- [21] GIBERT D, MATEU C, PLANES J, et al. Classification of malware by using structural entropy on convolutional neural networks[C]//*Proceedings of the AAAI Conference on Artificial Intelligence*. 2018, 32(1).
- [22] RAFF E, BARKER J, SYLVESTER J, et al. Malware detection by eating a whole exe[C]//*Proceedings of Workshops at the Thirty-second AAAI Conference on Artificial Intelligence*. 2018.
- [23] LE Q, BOYDELL O, MAC NAMEE B, et al. Deep learning at the shallow end: malware classification for non-domain experts[J]. *Digital Investigation*, 2018, 26: S118-S126.
- [24] GIBERT D, MATEU C, PLANES J. An end-to-end deep learning architecture for classification of malware's binary content[C]//*Proceedings of International Conference on Artificial Neural Networks*. 2018: 383-391.
- [25] YOUSEFI-AZAR M, VARADHARAJAN V, HAMEY L, et al. Autoencoder-based feature learning for cyber security applications[C]//*2017 International Joint Conference on Neural Networks (IJCNN)*. 2017: 3854-3861.
- [26] AHMADI M, ULYANOV D, SEMENOV S, et al. Novel feature extraction, selection and fusion for effective malware family classification[C]//*Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*. 2016: 183-194.
- [27] MCLAUGHLIN N, MARTINEZ DEL RINCON J, KANG B J, et al. Deep android malware detection[C]//*Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*. 2017: 301-308.
- [28] MAYS M, DRABINSKY N, BRANDLE S. Feature Selection for Malware Classification[C]//*MAICS*. 2017: 165-170.
- [29] RONEN R, RADU M, FEUERSTEIN C, et al. Microsoft malware classification challenge[J]. *ArXiv Preprint arXiv:1802.10135*, 2018.
- [30] GIBERT D. PE parser: a python package for portable executable

- files processing[J]. *Software Impacts*, 2022, 13: 100365.
- [31] 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.
- [32] BILAR D. Statistical structures: fingerprinting malware for classification and analysis[J]. *Proceedings of Black Hat Federal 2006*, 2006.
- [33] KIM Y. Convolutional neural networks for sentence classification[J]. *ArXiv Preprint arXiv:1408.5882*, 2014.
- [34] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[J]. *Advances in Neural Information Processing Systems*, 2017, 30.
- [35] AREVALO J, SOLORIO T, MONTES-Y-GOMEZ M, et al. Gated multimodal units for information fusion[J]. *ArXiv Preprint arXiv:1702.01992*, 2017.

[作者简介]



陈仲元（1998—），男，河南周口人，北京工业大学硕士生，主要研究方向为网络与信息安全、深度学习、可信计算。



张建标（1969—），男，江苏海门人，北京工业大学教授、博士生导师，主要研究方向为可信计算、系统安全、云安全。