文章编号：2096-1618(2025)01-0021-08

基于改进的图注意机制模型的安卓恶意软件检测方法研究

唐明婕， 甘 刚
(成都信息工程大学网络空间安全学院，四川 成都 610225)

摘要：在当下恶意软件蔓延的背景下，恶意软件检测需求不断增加。提出一种基于改进的图注意机制模型的安卓恶意软件检测方法，通过静态分析提取API调用图，显示出应用程序的行为。通过使用SDNE图嵌入算法，从API调用图中进行结构特征和内容特征的学习。在模型学习的过程中，采用一种计算双向图注意力权重的策略，旨在提高对相似节点的保留，并增强节点属性之间的相似性。最后，借助自注意力卷积层生成权重自适应的表示，并在池化层中生成图嵌入表示，以用于检测任务。基于CICMalDroid 2020数据集显示，该方法在安卓恶意软件检测领域表现出较高的有效性，准确率达到97.90%。与原有的图注意力网络模型相比，准确率提升0.03%，验证了该方法的实用性和有效性。该研究成果显示出该方法在应对不断增长的恶意软件威胁方面具有潜力，可为安卓恶意软件检测提供更准确和可靠的解决方案。

关 键 词：API调用图；SDNE嵌入；双向图注意力；安卓恶意软件检测
中图分类号：TP309.2 **文献标志码**：A
doi：10.16836/j.cnki.jcuit.2025.01.004

0 引言

Android系统是由谷歌公司开发，基于Linux内核而构建的具有开放性源代码操作系统。它是全球范围内最受欢迎的移动操作系统之一，拥有超过25亿的活跃用户，覆盖190多个国家，市场份额超过85%。Android的开源性和自由性吸引了众多开发人员积极参与应用软件开发，为用户提供了丰富多样的应用程序选择。

随着科技进步，Android系统面临着不断演变和普及的恶意软件威胁。恶意软件是一种计算机程序，其目的是在没有经过授权的情况下执行危害性操作，如病毒、蠕虫和特洛伊木马，这些恶意程序通过破坏正常的软件运行来获取对计算机系统的控制权。恶意软件不断增长，涌现出各种新型恶意软件，如勒索软件可以锁定用户的文件并勒索赎金。此外，还有针对移动平台和物联网设备的恶意软件威胁。随着智能手机和平板电脑的广泛普及，移动平台已成为恶意软件攻击的新兴目标。恶意软件攻击者越来越多地利用社交工程技术和社交媒体平台来诱骗用户。通过伪造身份、钓鱼链接、欺诈性信息等手段诱使用户点击恶意链接或共享敏感信息。恶意软件针对于Android系统也存在许多威胁，包括数据泄露和窃取用户敏感信息问题、通过监视和截取用户的交易信息，或模拟银行应用程序来进行欺诈性交易问题、勒索软件、在用户的设备上显示大量的广告或发送垃圾短信和电子邮件、控制用户的设备，使攻击者能够监视用户的活动、偷窥个人信息或进行非法操作。鉴于Android系统是全球最广泛使用的移动操作系统，成为恶意软件攻击的主要目标，因此研究和发展Android恶意软件检测技术变得极为重要和紧迫。

收稿日期：2023-09-13
基金项目：四川省科技计划资助项目(23ZDYF0380、2021ZYD0011)
通信作者：甘刚. Email：test_me@cuit.edu.cn

1 相关工作

机器学习驱动的Android恶意软件检测方法可分为两大类：静态检测和动态检测。静态检测提取特征的效率高、可扩展性好，同时也保护了用户的隐私。然而，静态检测无法捕获动态行为，处理复杂代码，且容易出现漏报误报的问题。相比之下，动态检测能够提取特征准确性高，但它的开销较大。

1.1 静态检测

静态检测[1]是在应用程序不运行的情况下，通过反编译APK文件来提取应用程序权限、Java代码、意图、字符串等特征进行分析和检测。秦中元等[2]提出一种静态综合检测方法，利用MD5消息摘要值来快速匹配已经检测过的APK文件，并运用污点传播和语义分析技术来检测未经过检测的APK文件。张锐[3]针对恶意倾向进行初步判断，通过权限之间的相关性选取和去除冗余特征属性来构建代表性的权限特征集合，并作为分类依据进行实验，但仅以Android权限作

为依据并不具备代表性;另外,为解决静态检测在新型恶意软件检测方面精确性下降的问题提出一种方法,通过采用从逆向工程获得的高风险 API 调用作为检测特征,并运用概率神经网络进行分类,以提高检测的实时性和泛化性能。然而,要获得更出色的检测效果,仅仅依赖静态检测是不够的。王站[4]提出一个 Android 平台恶意代码检测框架,主要侧重于静态检测,研究对象是 dex 文件,而关键特征是高风险 API 调用。此外,该框架使用 TF-IDF 算法将恶意样本抽象为文本表示,然后通过相似度计算、后台监听以及黑名单匹配等技术来进行恶意代码的检测。这种方法的目的是提高 Android 平台上恶意代码的识别和检测效果。王军等[5]利用多类特征进行恶意软件分类的方法,通过统计分析构建恶意软件家族特征库,提出恶意软件家族特征综合描述方法。为应对由代码混淆技术引发的问题,Tian 等[6]提出一种基于代码异构性分析的新型 Android 重新打包的恶意软件检测技术,将应用程序与用户之间的交互、敏感 API、权限作为特征,评估 4 种机器学习技术进行恶意软件分类。与传统方法相比,这种方法减少了对程序代码的依赖性,提高了检测的准确性和效率。但由于对应用程序进行全面的特征表示,因此,检测效率可能会相对较低。相同地,Fan 等[7]提出通过敏感的子图分析检测安卓系统附带的应用程序的方法,对敏感的 API 节点进行特征提取,构建敏感子图,并采用 4 种分类器(随机森林、决策树、KNN、PARt)来实施对恶意应用程序的检测。这些技术的优点在于能减少对代码特征的依赖,提高检测的准确性,以及增强对代码混淆攻击技术的抵御能力。然而,这种方法可能存在一定局限性,因为它可能难以有效地识别采取了加密攻击技术的恶意应用程序。

1.2 动态检测

动态检测[8]是指当程序处于运行状态时,收集系统的信息并建立特征库,执行相似性对比,以实现应用程序的检测。在应用程序运行时,除了系统调用信息之外,还能够利用其他信息来对应用程序进行特征描述。Gianazza 等[9]提出 PuppetDroid 动态分析检测系统,通过实时检测用户与应用程序的 UI 界面的交互行为来自动触发恶意行为。这种方法的目标是实现对恶意应用的检测,而不仅限于依赖系统调用信息。类似地,Shabtai 等[10]提出 Andromaly 检测系统,利用收集应用程序运行时的信息作为特征,通过有监督分类器的训练,能够快速检测出恶意应用程序。这种方式超越了仅仅依赖特定类型的信息,而是综合考虑了应用程序在运行时的多个方面。Saracino 等[11]提出 MADAM 检测系统,采用多层面的方法对应用程序进行特征表示,包括内核层面、应用层面和用户层面。随后,它通过特征融合和应用程序关联分析的方式来实现对恶意应用的检测。这种多层面的特征表示和分析方法有助于提高恶意应用检测的效率和准确性。彭守镇[12]提出了一种基于模糊神经网络的恶意应用程序动态检测技术。该方法提取应用程序的权限及代码作为特征,并构建模糊神经元及动态模糊神经网络来进行特征匹配,最后生成相应的风险提示报告,准确率为 87.35%。传统方法无法充分描述应用程序功能的完整流程,尤其在应用数据的特征提取方面。为此,Enck 等[13]提出了一个信息流跟踪系统 TaintDroid。该系统可以同时监控多个敏感数据源,并通过 4 个层次(变量级、消息级、方法级和文件级)进行监控,但无法追踪隐式数据流。基于这一基础,Zhang 等[14]引入了 VetDroid。这个动态分析系统可以从全新的角度(如敏感内容、系统资源调用)对恶意应用的敏感行为进行重新构建,进行更细粒度的分析,实现了对恶意应用的检测。通过分析隐私权限图,VetDroid 可以有效地发现恶意应用程序可能存在的隐私数据滥用问题。Platzer 等[15]提出的 AppsPlayground 检测系统,结合多种动态分析方法对应用程序进行检测分析,在检测隐私泄露和恶意应用行为方面展现了出色的效能。

1.3 基于图的检测

为获得更多样本的结构信息并进行多维度分析,研究者们越来越倾向于采用基于图的检测方法,如图神经网络[16](graph neural network,GNN)、图卷积神经网络[17](graph convolution networks,GCN)及图注意力网络[18](graph attention networks,GAT)。这些方法相比传统的机器学习和深度学习方法,能够提供更高的性能。Busch 等[19]通过监测应用程序执行过程中产生的网络流量数据构建有向图,并使用 GNN 对恶意软件进行分类。为避免运行时开销,Feng 等[20]通过从 Android 应用程序的函数调用关系构建近似的调用图表示应用程序,并进一步提取内部属性,然后使用 GNN 进行恶意软件识别。Li 等[21]利用 GCN 训练并取得 98.32% 的检测准确率。相较于 GCN,GAT 采用注意力机制来替代传统的卷积操作,更有效地将邻居节点的特征聚合到中心节点上。此外,图注意力网络在多个领域都得到了广泛的应用。Wu 等[22]将双图注意力网络用于推荐系统中多层面社会效应的深层潜在表示,实验证明该模型在推荐精度上取得了很大提升。Wang 等[23]利用自注意力权重和先验学习的类共现知识,提出新的权重表示方法,实验证明,该方法可以建

模依赖关系,提高分类性能。Qiu 等[24]提出面向癌症预测的门控图注意力网络(GGAT),能够充分挖掘相关样本之间的潜在相关性,实验证明在癌症预测任务上的精度提高了 1% ~2%。

这些研究表明基于图的检测方法在各个领域具有广泛的应用,并取得了显著的改进和成果。不断的探索和创新有助于进一步提高图神经网络的性能和适应性。

2　方法

2.1　概述

本文介绍一个用于安卓恶意软件检测的方法,包括以下 5 个步骤:(1)APK 源代码提取:使用反编译工具提取 APK 源代码文件;(2)API 调用信息提取和转换:提取 APK 中的 API 调用信息,并将其转换成 API 调用图的形式;(3)节点嵌入向量提取:利用节点嵌入算法提取 API 调用图中节点的嵌入向量表示;(4)训练集构建:将提取的节点嵌入向量构建成训练集;(5)DGAT 网络训练和分类:将构建好的训练集输入 DGAT 网络进行训练,并在测试集上进行良性和恶性样本的分类。流程如图 1 所示。

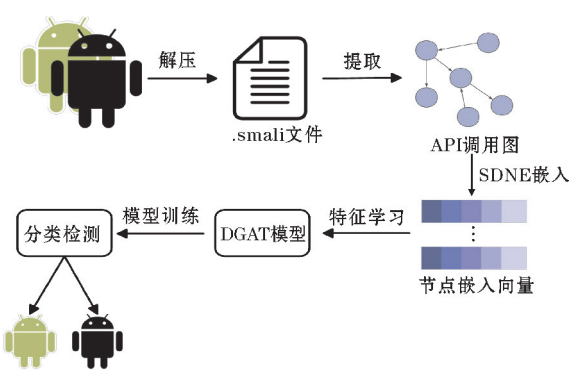

图 1　检测方法流程

2.2　提取 API 调用图

首先,使用解压缩工具(如 Breezip)从 APK 文件中提取. dex 文件。然后,运用反编译工具将. dex 文件还原为. smal 文件,以获取 Dalvik 字节码的源代码表示形式。接着,编写自动化程序,从反编译后的. smal 代码中提取相应的 API 调用。利用第三方工具库 GraphViz,将提取到的 API 调用构建成. dot 格式的有向图。这个有向图用于表现 APK 文件中不同部分之间的 API 调用关系。

2.3　SDNE 嵌入

图结构展现复杂网络的特征,如入侵检测、文本分析和推荐系统等应用。直接分析图结构耗时且耗空间,图嵌入方法应运而生。该方法将复杂的图形数据转化为低维向量表示,方便后续分析和应用。它能够将无向图、有向图以及带属性的图转换为低维向量集,然后将这些向量输入到机器学习或深度学习网络中,以执行复杂的检测和分类任务。DeepWalk[25]采用随机游走策略捕捉节点的局部上下文信息,以学习节点的表示向量。当两个节点在图中共享更多邻近节点(或高阶邻近节点)时,对应的向量之间的距离越接近。然而,随机游走方法存在一定局限性,难以有效地保留节点的局部关系。Node2vec[26]是对 DeepWalk 的改进,它引入了偏向性随机游走策略来生成节点序列,并使用 skip-gram 模型进行训练。Node2vec 通过手动调节参数 p 和 q,控制游走的偏向性,可以选择偏向于广度优先搜索(BFS)或深度优先搜索(DFS)。通过对 DeepWalk、Node2vec 和 SDNE 等 3 种图嵌入方法的评估,发现 SDNE 在处理数据稀疏和全局关联问题时表现较好,可以获得更高的准确率。因此,本文选择 SDNE 算法作为图嵌入的方法。

SDNE(structural deep network embedding)[27]是一种基于深度神经网络和自编码器的图嵌入算法,目标是通过学习图的结构信息,将图中的节点映射到低维向量空间中。SDNE 由两个主要模块组成:无监督和有监督部分。无监督部分包括一个深度自编码器,用于学习节点的二阶相似度信息,而有监督部分则使用拉普拉斯特征映射来捕获节点的一阶相似度。在使用 SDNE 之前,需要对图结构信息进行编码,生成相应的邻接矩阵。

定义 1 (一阶相似度)对于任何一对顶点,记作 v_i、v_j,定义它们之间的一阶相似度为 $s_{i,j}$,如果 $s_{i,j}>0$,那么说明顶点 v_i 和 v_j 之间存在一阶相似性。否则,它们之间的一阶相似度为 0。

定义 2 (二阶相似度)设已经计算了节点 v_u 与其他所有顶点之间的一阶相似度,记为 $N_u=\{s_{u,1},\cdots,s_{u,|v|}\}$。那么,节点 u 和节点 v 之间的二阶相似度可以根据上述的一阶相似度来计算。

通过上述定义可知,一阶相似度具有边相连的节点的 Embedding 向量具有相似性,主要反映了 Graph 的局部特征;二阶相似度拥有共同邻居但不是直接相连的两个顶点之间应该具有相似性,反映了 Graph 的全局特征。SDNE 的拉普拉斯映射模块承担计算一阶相似度的任务,具体来说,如果两个顶点 v_i 和 v_j 之间存在一条边,那么它们的节点嵌入映射结果将会很相似。这种相似性可以通过一阶相似度的损失函数来衡量:

$$\mathcal{L}_{1st}=\sum_{i,j=1}^{n}S_{i,j}\|y_i^{(K)}-y_j^{(K)}\|_2^2=\sum_{i,j=1}^{n}S_{i,j}\|y_i-y_j\|_2^2 \quad (1)$$

自编码器模块在 SDNE 中承担处理二阶相似度的任务。模型的输入数据是图的邻接矩阵,这个矩阵本质上包含了每个节点 i 的邻接结构信息。因此,通过输入这个矩阵,模型可以学习到与节点 i 相邻的节点的表示,这些表示可以用来捕捉节点之间的二阶相似度。模型通过不断优化代价函数来学习全局结构特征,以便更好地嵌入节点。然而,需要注意的是,由于图在实际应用中通常是稀疏的,邻接矩阵中的零元素远远多于非零元素。使用邻接矩阵作为输入要处理很多零,会降低非零元素的重要性,为避免此情况发生,对损失函数进行优化,优化后的二阶相似度损失函数如式(2)所示。

$$\mathcal{L}_{2nd}=\sum_{i=1}^{n}\|(\hat{x}_i-x_i)\odot b_i\|_2^2=\|\hat{X}-X\odot B\|_F^2 \quad (2)$$

式中:$\odot$ 是哈马达乘积,表示元素相乘;$b_i=\{b_{i,j}\}_{j=1}^{n}$,若 $s_{i,j}=0$,则 $b_{i,j}=1$,否则,$b_{i,j}=\beta>1$。这种方式可以用来强调存在边连接的节点对之间的关系,赋予它们更高的重要性,同时对没有边连接的节点对进行权重调整,以反映它们之间的关系较弱。

2.4 改进的图注意机制模型 DGAT

GAT 通过引入注意力机制来在信息传递过程中考虑邻居节点的不同贡献度,以便更好地捕捉节点之间的关系,这允许模型学习节点之间的注意力权重,从而有效地聚合领域信息。图 2 为单层图注意层中心节点特征的聚合方式。

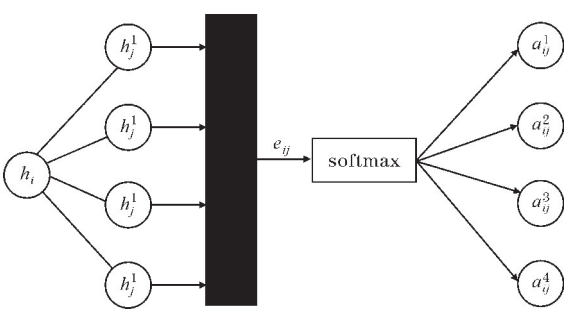

图 2 GAT 网络中心节点聚合方式

在 GAT 模型训练中,节点特征 $h=\{\vec{h_1},\vec{h_2},\cdots,\vec{h_N}\}$ 被集合为组,$\vec{h_i}\in\mathbb{R}^F$,其中 N 代表节点的数量,F 表示每个节点的特征维度。对于每个节点,共享一个权重矩阵来执行线性变换:$W\in\mathbb{R}^{F'\times F}$,再在每个节点上使用共享注意机制 $\alpha:\mathbb{R}^{F'}\times\mathbb{R}^{F'}\to\mathbb{R}$ 为每条边 (i,j) 计算贡献度权重,式(3)这个值表明了节点 j 的特征对节点 i 的重要性:

$$e(\vec{h_i},\vec{h_j})=\mathrm{LeakyReLU}(\alpha^{\mathrm{T}}\cdot[W\vec{h_i}\|W\vec{h_j}]) \quad (3)$$

为确保在不同节点之间进行比较时系数具有可比性,采用 softmax 函数对 $j\in N_i$ 的权重进行归一化处理来定义注意力系数:

$$\alpha_{ij}=\mathrm{softmax}_j(e(\vec{h_i},\vec{h_j}))=\frac{\exp(e(\vec{h_i},\vec{h_j}))}{\sum_{j\in N_i}\exp(e(\vec{h_i},\vec{h_j}))} \quad (4)$$

归一化注意系数被用于计算每个节点对应的特征的线性组合,这个线性组合的结果被用作每个节点的最终输出特征:

$$\vec{h_i'}=\sigma(\sum_{j\in N_i}\alpha_{ij}\cdot W\vec{h_j}) \quad (5)$$

DGAT 是对 GAT 模型的改进,引入双重的图注意力机制,以更好地捕捉节点之间的关系。它由两个核心模块组成:节点级双向注意的图卷积模块和图级自注意的图池化模块。DGAT 通过双重的图注意力机制同时考虑了节点级别和图级别的关系,以更好地建模复杂的图结构和节点关联性。图 3 为改进的图注意机制模型的结构。

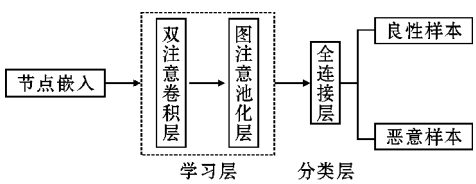

图 3 改进的图注意机制模型结构

2.4.1 节点级双向注意图卷积

DGAT 模型采用一种称为双向图注意力机制的图注意力机制,以加强节点表示特征的学习,从而更好地聚合节点邻域信息特征。这种机制可以同时考虑节点的邻居节点对其表示的影响,以及节点对邻居节点的影响。图 4 为双向图注意力机制的示意图。

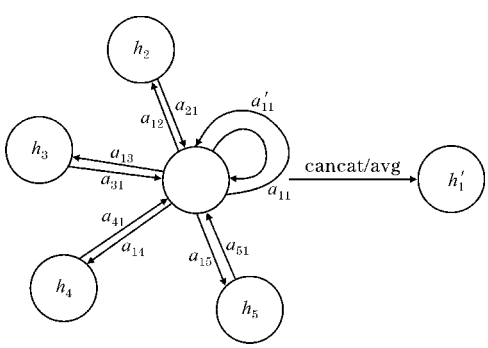

图 4 双向图注意力机制

双向图注意力机制通过引入双向的注意力权重,更好地捕捉节点与其邻居之间的关系。相较于传统的单向图注意力机制,双向图注意力机制能够同时考虑节点的输入特征和其邻居节点的特征,从而更准确地聚合邻域信息。使用静态和动态的注意力权重计算对节点输入特征进行加权,然后将它们应用于节点表示

中。DGAT 模型能够更精确地表达节点之间的依赖和关联关系,从而提高了特征学习的性能和效果。

注意机制 α 的实现,是通过一个神经网络层来构建的,这个神经网络层的权重矩阵 $a \in R^{2F'}$ 会在训练过程中进行学习和优化,以确保注意力权重总和为 1。此外,LeakyReLU 作为激活函数。在对所有邻居节点的相关度归一化系数进行计算后,通过这些系数对相应的特征线性组合进行计算,这将构成每个节点的最终输出特征,关于节点 i 以及其邻居节点 j 之间的正向注意力系数:

$$\alpha_{ij}=\frac{\exp(\mathrm{LeakyReLU}(\alpha^{\mathrm{T}}\cdot[W\overrightarrow{h_i}\|W\overrightarrow{h_j}]))}{\sum_{k\in N_i}\mathrm{LeakyReLU}(\alpha^{\mathrm{T}}\cdot[W\overrightarrow{h_i}\|W\overrightarrow{h_k}])} \tag{6}$$

计算反向注意力 α_{ji} 时,以节点 j 作为中心节点,对节点 i 进行归一化,结合式(4)计算 $\mathrm{softmax}_i$。反向注意力 α_{ji}:

$$\alpha_{ji}=\mathrm{softmax}_i(e(\overrightarrow{h_i},\overrightarrow{h_j}))=\frac{\exp(e(\overrightarrow{h_i},\overrightarrow{h_j}))}{\sum_{j\in N_i}\exp(e(\overrightarrow{h_i},\overrightarrow{h_j}))} \tag{7}$$

最终双向注意系数为

$$\alpha_{ij}^{\mathrm{final}}=\alpha_{ij}+\alpha_{ji} \tag{8}$$

2.4.2　图级自注意图池化

图级自注意图池化部分是一种将子注意力机制引入图池化过程中的方法,它着重关注了图中包含的丰富特征信息以及节点在图结构中的位置。首先通过使用图卷积计算自注意权重,以便输入特征能够在权重计算中起到关键作用。目标是决定哪些节点的信息应该被保留,哪些节点的信息应该被丢弃。图级自注意池化层通过自动生产的注意力分数 $Z \in R^{N\times 1}$ 来选择高权重的节点特征,以生成图的有效表征。计算公式如下:

$$Z=\delta(\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}X\theta_{\mathrm{att}}) \tag{9}$$

式中,δ 是激活函数,$\widetilde{\boldsymbol{A}} \in R^{N\times N}$ 是具有自连接的邻接矩阵 $\widetilde{\boldsymbol{A}}=A+I$,$\widetilde{\boldsymbol{D}} \in R^{N\times F}$ 是 $\widetilde{\boldsymbol{A}}$ 的度矩阵,$X \in R^{N\times N}$ 表示该图包含 N 个节点以及 F 维特征,在图级自注意图池化层中,参数 $\theta_{\mathrm{att}} \in R^{F\times 1}$ 的设置影响着每个节点的权重分配,决定了它们在图级任务中的相对重要性。在池化层的作用下,每个节点根据这些评分被筛选,保留重要信息用于构建新的、粗化的图拓扑结构。

3　实验及结果

3.1　实验环境与数据集

使用的恶意软件样本来自 CICMalDroid 2020 数据集[28],该数据集是 CICDataset 的一个子数据集。CICMalDroid-2020 数据集由加拿大通讯与信息技术安全研究实验(canadian institute for cybersecurity)于 2020 年创建。该数据集包含超过 17341 个 Android 恶意样本,涵盖广告软件、银行恶意软件、短信恶意软件、风险软件和正常样本 5 个不同类别的样本。采用 CICMalDroid-2020 数据集,从中随机选取了 2000 个恶意样本,并分别从 Google 应用商店和 Drebin 数据集中各选取 1000 个良性样本。每个样本都经过标准的检查,以确保有效性,旨在保证所选的恶意和良性样本在研究中的准确性和可靠性。为进行模型的性能评估,从总体样本中随机挑选 400 个样本作为测试集。数据集样本的分布如表 1 所示。

表 1　数据集样本分布

样本类别	训练样本集	测试样本集
恶意样本	2000	400
良性样本	2000	400

硬件环境采用 Intel Core CPU i5-1135 G7 处理器,16GB 内存,PTX2080 GPU。软件环境采用 cuda12.1,python3.10。

3.2　评估指标

为评估实验的性能和效果,采用 4 个重要的指标来进行评估,指标包括准确率(Accuracy)、精确率(Precision)、召回率(Recall)和 F1 分数(F1-Score)。指标的计算依赖 4 个关键的值,即真正例(true positives,TP)、真负例(true negatives,TN)、假正例(false positives,FP)和假负例(false negatives,FN)。

(1)准确率:表示正确预测的样本占总样本数量的比例

$$\mathrm{Accuracy}=\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$$

(2)精确率:表示在所有被分类为正例的样本中,有多少是真正例

$$\mathrm{Precision}=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$$

(3)召回率:表示所有真正例中有多少被正确地分类为正例

$$\mathrm{Recall}=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$$

(4)F1 分数:综合考虑了精确率和召回率,是一个综合性能指标

$$\mathrm{F1\text{-}Score}=\frac{2\cdot\mathrm{Precision}\cdot\mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}}$$

3.3 对比实验

3.3.1 图嵌入长度评估实验

图嵌入算法的主要目标是将复杂的图形数据映射到向量空间中,以进一步的分析和处理。映射过程中,嵌入长度是一个关键参数,它决定了生成的特征向量的维度。嵌入长度的选择对于最终的特征表示非常重要。如果嵌入长度设置过低,可能会导致信息的损失,因为无法充分捕捉图的复杂结构和关系。反之,如果嵌入长度过高,可能会导致向量维度过多,容易引发过拟合问题。为找到最适合的嵌入长度,进行一系列实验,从嵌入长度为 16 开始,以 8 为步长逐渐增加,最长测试到 40。根据不同嵌入长度得到的检测结果如表 2 所示。由表 2 可知,在图嵌入长度为 32 时,得到最合适的性能表现。

表 2 不同嵌入长度的检测结果 单位:%

嵌入长度	Accuracy	Precision	Recall	F1-Score
16	94.24	95.18	94.89	95.18
24	95.58	95.57	95.23	95.31
32	96.07	95.70	95.41	95.45
40	96.17	95.73	95.47	95.49

3.3.2 Dropout 率评估实验

Dropout 是一种被用来应对神经网络中过拟合问题的技术。为找到最适合的 Dropout 率,进行一系列实验,比较在不同的 Dropout 率下训练得到的模型的性能。不同 Dropout 率的检测结果如表 3 所示,从表 3 可知,当 Dropout 率设置为 0.3 时,模型表现出最佳的检测效果。

表 3 不同 Dropout 率的检测结果 单位:%

Dropout 率	Accuracy	Precision	Recall	F1-Score
0.1	95.31	95.28	95.23	95.56
0.2	96.26	96.68	96.12	96.72
0.3	97.34	97.03	97.06	97.11
0.4	97.25	96.78	97.05	96.89

3.3.3 模型对比实验

Xiao 等[29]使用一种动态分析方法,通过捕获恶意软件执行过程中的系统调用序列来提取特征,运用长短时记忆神经网络(LSTM)进行检测;Catal 等[30]从恶意和良性 APK 中提取 API 调用图,利用 Nodevec 技术生成嵌入向量,将这些向量输入到 GCN 和 GAT 中进行检测;Yue 等[31]利用 AMD 数据集提取 API 调用图,使用 SDNE 生成嵌入向量,输入 GAT 进行检测。从表 4 可知,本文方法在准确率、精确度、召回率和 F1 分数方面表现最优,具有更好的性能,因此在安卓恶意软件检测方面效果更佳。

表 4 不同模型性能对比 单位:%

检测方法	Accuracy	Precision	Recall	F1-Score
LSTM	93.10	94.76	91.31	93.00
GAT 和 GNN	96.10	95.21	94.39	94.80
GAT	97.87	97.34	97.47	97.40
本文方法	97.90	97.38	97.51	97.43

4 结论

本文使用 API 调用图提取恶意软件特征,通过 SDNE 嵌入方法生成节点嵌入向量,使用改进的图注意机制模型 DGAT 进行安卓恶意软件检测实验。结论如下:

(1)使用本文的方法能够提高安卓恶意软件检测的准确率。

(2)改进的图注意机制模型相比原始的图注意力网络模型表现更好,准确率达97.90%,比原模型提高 0.03%。

(3)本文仅使用静态检测方法,未来的研究可以将恶意软件的行为特征考虑进去,以达到更好的检测效果。

参考文献:

[1] Chen H M,JiangCun H U. Static Detection Method of Android Malware[C]. Computer Systems & Applications,2018.

[2] 秦中元,徐毓青,梁彪,等. 一种 Android 平台恶意软件静态检测方法[J]. 东南大学学报(自然科学版),2013,43(6):1162-1167.

[3] 张锐. Android 环境下恶意软件静态检测方法研究[D]. 重庆:重庆大学,2014.

[4] 王站. Android 平台恶意代码静态检测技术的研究与实现[D]. 成都:电子科技大学,2014.

[5] 王军,庄毅,潘家晔. 一种 Android 恶意软件多标签检测方法[J]. 小型微型计算机系统,2017,38(10):2307-2311.

[6] Tian K,Yao D,Ryder B G,et al. Detection of Repackaged Android Malware with Code-Heterogeneity Features[J]. IEEE Transactions on Dependable

and Secure Computing,2020,17(1):64-77.

[7] Fan M, Liu J, Wang W, et al. DAPASA: Detecting Android Piggybacked Apps Through Sensitive Subgraph Analysis[J]. IEEE Transactions on Information Forensics and Security,2017,12(8):1772-1785.

[8] Jamalpur S, Navya Y S, Raja P, et al. Dynamic Malware Analysis Using Cuckoo Sandbox[C]. Second International Conference on Inventive Communication & Computational Technologies,2018:1056-1060.

[9] Gianazza A,Maggi F,Fattori A,et al. PuppetDroid: A User-Centric UI Exerciser for Automatic Dynamic Analysis of Similar Android Applications[EB/OL]. Https://arxiv. org/abs/1402. 4826,2014.

[10] Shabtai A,Kanonov U,Elovici Y,et al. "Andromaly": a behavioral malware detection framework for android devices[J]. Journal of Intelligent Information Systems,2012,38(1):161-190.

[11] Saracino A, Scandurra D, Dini G, et al. MADAM: Effec-tive and efficient behavior-based Android malware detectionand prevention[J]. IEEE Transactions on Dependable andSecure Computing,2018,15(1):83-97.

[12] 彭守镇. 基于模糊神经网络的恶意 APP 软件动态检测技术研究[J]. 现代电子技术,2020,43(2):49-52.

[13] Enck W,Gilbert P,Cox L P,et al. TaintDroid: An Information Flow Tracking System for Real-Time Privacy Monitoring on Smartphones[J]. Communications of the ACM,2014,57(3):99-106.

[14] Zhang Y,Yang M,Xu B,et al. Vetting undesirable behaviors in android apps with permission use analysis[C]. Computer and Communications Security. ACM,2013.

[15] Platzer C, Lindorfer M, Neugschwandtner M, et al. ANDRUBIS-1,000,000 Apps Later: A View on Current Android Malware Behaviors[C]. Third International Workshop on Building Analysis Datasets & Gathering Experience Returns for Security, IEEE,2016.

[16] F Scarselli, M Gori, A C Tsoi, et al. The Graph Neural Network Model[J]. IEEE Transactions on Neural Networks,2009,20(1):61-80.

[17] Kipf T N,Welling M. Semi-Supervised Classification with Graph Convolutional Networks[EB/OL]. Https://arxiv. org/abs/1609. 02907,2016.

[18] Petar Veli kovi,Guillem Cucurull,Arantxa Casanova,et al. Graph Attention Networks[EB/OL]. Https://arxiv. org/abs/1710. 10903,2017.

[19] Busch J,Kocheturov A,Tresp V,et al. NF-GNN: Network Flow Graph Neural Networks for Malware Detectionand Classification [EB/OL]. Https://arxiv. org/abs/2103. 03939,2021.

[20] Feng P,Ma J,Li T,et al. Android Malware Detection via Graph Representation Learning[J]. Mobile Information Systems. 2021;2021(6):1-14.

[21] Li S,Zhou Q,Zhou R,et al. Intelligent malware detection based on graph convolutional network [J]. The Journal of Supercomputing, 2021;78(4): 4182-4198.

[22] Wu Q,Zhang H,Gao X,et al. Dual Graph Attention Networks for Deep Latent Representation of Multifaceted Social Effects in Recommender Systems [EB/OL]. Https://arxiv. org/abs/1903. 10433,2019.

[23] Wang H ,Zhao W,Li Z,et al. A Weighted Graph Attention Network Based Method for Multi-label Classification of Electrocardiogram Abnormalities [C]. 2020 42nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC) in conjunction with the 43rd Annual Conference of the Canadian Medical and Biological Engineering Society. IEEE,2020.

[24] Qiu L,Li H,Wang M,et al. Gated Graph Attention Network for Cancer Prediction[J]. Sensors, 2021,21(6): 1938.

[25] Perozzi B, AlFRfou R, Skiena S. Deepwalk: online learning of social representations [C]. Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. New York: ACM,2014.

[26] Grover A, Leskovec J. Node2vec: scalable feature learning for networks [C]. Proceedings of the 22nd ACM SIGKDD International conference on Knowledge discovery and data mining. San Francisco: ACM,2016.

[27] Wang D, Peng C, Zhu W. Structural deep network embedding [C]. Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. San Francisco:

ACM,2016.
[28] Garcia J, Sillero A, Montes M, et al. CICMALDROID2020: a benchmark dataset for Android malware detection based on mimicry and evasion techniques[J]. Future Generation Computer Systems,2020,109:420-435.
[29] Xiao X, Zhang S, Mercaldo F, etal. Android malware detection based on system call sequences and LSTM[J]. Multimed Tools Appl,2019,78:3979.
[30] Catal C, Gunduz H, Ozcan A. Malware detection based on graph attention networks for intelligent transportation systems[J]. Electronics,2021,10(20): 2534.
[31] Yue Z W, Fang Y, Zhang. Android malware detection based on graph attention networks[J]. J Sichuan Univ: Nat Sci Ed,2022,59:053002.

Research on Android Malware Detection Method based on the Improved Graph Attention Mechanism Model

TANG Mingjie, GAN Gang
(College of Cybersecurity, Chengdu University of Information Technology, Chengdu 610225, China)

Abstract: In the context of the spread of malware, the demand for malware detection is increasing. This paper presents an Android malware detection method based on an improved graph attention mechanism model. The methodology involves extracting the API call graph through static analysis, which shows the behavior of the application. Subsequently, the structural features and content features are acquired from the API call graph by using the SDNE graph embedding algorithm. In the process of model learning, a strategy is adopted to calculate the bidirectional graph attention weights, aiming to improve the retention of similar nodes and enhance the similarity between node properties. Finally, a weight-adaptive representation is generated with the help of the self-attention convolution layer, and a graph embedding representation is generated in the pooling layer for use in the detection task. The experimental results are based on the CICMalDroid 2020 data set, showing that this method shows high effectiveness in the field of Android malware detection, with an accuracy of 97.90%. Compared with the original graph attention network model, it improves the accuracy by 0.03%, verifying the practicability and effectiveness of the proposed method. The results show the potential to deal with growing malware threats and to provide a more accurate and reliable solution for Android malware detection.

Keywords: API call graph; SDNE embedding; bidirectional graph attention; Android malware detection