基于图注意力网络的安卓恶意软件检测

岳子巍，方　勇，张　磊
(四川大学网络空间安全学院，成都 610065)

摘　要：安卓恶意软件的爆发式增长对恶意软件检测方法提出了更高效、准确的要求. 早年的检测方法主要是基于权限、opcode 序列等特征，然而这些方法并未充分挖掘程序的结构信息. 基于 API 调用图的方法是目前主流方法之一，它重在捕获结构信息，可准确地预测应用程序可能的行为. 本文提出一种基于图注意力网络的安卓恶意软件检测方法，该方法通过静态分析构建 API 调用图来初步表征 APK，然后引入 SDNE 图嵌入算法从 API 调用图中学习结构特征和内容特征，再通过注意力网络充分融合邻居节点特征向量，进而构成图嵌入进行检测任务. 在 AMD 数据集上的实验结果表明，本文提出的方法可以有效检测恶意软件，准确率为 97.87%，F_1 分数为 97.40%.

关键词：安卓恶意软件；图注意力网络；API 调用图；图嵌入
中图分类号：TP391.1　　**文献标识码**：A　　DOI：10.19907/j.0490-6756.2022.053002

Android malware detection based on graph attention networks

YUE Zi-Wei, *FANG Yong*, *ZHANG Lei*
(School of Cyber Science and Engineering, Sichuan University, Chengdu 610065, China)

Abstract: The explosive growth of Android malware has put forward more efficient and accurate requirements for malware detection methods. In the early years, detection methods were mainly based on features such as permissions and opcode sequences. However, these methods did not fully mine the structural information of programs. The method based on API call graph is one of the mainstream methods. It focuses on capturing structural information and can accurately predict the possible behavior of the application. This paper proposes an Android malware detection method based on graph attention network. The method constructs an API call graph through static analysis to initially characterize the APK, and then introduces the SDNE graph embedding algorithm to learn structural and content features from the API call graph. The attention network fully fuses the neighbor node feature vectors, and then forms the graph embedding for the detection task. The experimental results on the AMD dataset show that the proposed method can effectively detect malware with an accuracy of 97.87% and an F_1 score of 97.40%.

Keywords: Android malware; Graph attention network; API call graph; Graph embedding

1　引　言

安卓系统早已占据智能移动设备市场中的主要份额，其开源、便捷的特性受到了广大开发者和用户的青睐. 然而用户在享受应用程序所提供的丰富服务时，也时刻面临着安全威胁. 360 安全实验室发布的《2021 年中国手机安全状况报告》[1] 数据显示，2021 年全年新增移动端恶意程序样本约 943

收稿日期：2022-03-07
作者简介：岳子巍 (1997—)，男，甘肃兰州人，硕士研究生，主要研究领域为 Android 恶意代码检测.
通讯作者：张磊. E-mail：zhanglei2018@scu.edu.cn

万个,平均每天新增 2.58 万个.安卓恶意软件可在用户尚未感知或未经用户同意的情况下运行,一般而言恶意软件具有以下一种或多种行为:恶意扣费、窃取和修改用户数据、收集用户信息、恶意捆绑和勒索等其他恶意行为.这些行为会严重侵犯用户的合法权益,甚至会造成个人信息泄露和财产损失.因此,有效的安卓恶意软件检测技术对于减少用户财产损失和隐私泄露具有重要意义.

目前主流的安卓恶意软件主要有两类检测方法:基于签名的检测和基于行为的检测.基于签名的检测方法无需执行软件,借助反编译技术,从语法和语义的角度分析软件源码或二进制代码,一般具有检测速度快、误报率低的优点.然而基于签名的检测方法无法有效地检测出经过重打包、加密和变形等混淆技术处理后的恶意软件,面对未知恶意软件时也较为乏力.基于行为的检测方法原理是在沙箱中执行待检测的软件,并从运行过程中获取并记录如 API 调用、文件操作和网络操作等运行时行为特征,再从这些行为上判定是否为恶意软件.基于行为的检测方法普遍时间开销大,对实时性要求高,且在虚拟环境中运行软件时,程序无法精准模拟人类操作,难以使软件遍历所有可能的执行路径,尤其是面对带有虚拟环境对抗的软件时这种劣势更为明显.

为更好地表示恶意软件的特征,提高恶意软件检测的效果,本文提出一种基于图注意力网络的安卓恶意软件检测方法:首先,对安卓应用进行反编译,提取 smali 代码;然后从 smali 代码中提取 API 调用,构造为图,再根据预设的阈值筛选出边数符合要求的 API 调用图;接下来借助图嵌入算法,将处理后的 API 调用图转换为包含图中有效信息的向量表示;最后将向量输入到 GAT 网络中进行训练和检测.本文主要的研究工作与贡献包括:(1)利用静态分析构建出 API 调用图,以更好地表征 APK 的结构语义,图中节点表示 API 函数,边表示 API 之间的调用关系,通过 API 调用的内容和序列分析程序的执行路径;(2)提出一种新颖的基于图注意力网络的安卓恶意软件检测方法,将 SDNE 与 GAT 相结合,可从图中节点挖掘出更丰富的信息.

2　相关工作

2.1　基于签名的检测

基于签名的检测方法是一种重要的检测方法,其原理是通过对恶意软件进行静态分析,分析出恶意软件独特的信息(如权限、API、汇编代码或二进制代码特征),并将这些信息总结为签名特征数据库,在已知恶意软件的签名特征的情况下,通过与待检测软件的签名进行匹配,如在特征数据库中找到相同的签名特征则将待检测软件判定为恶意软件,否则判定为良性软件.常见的签名分析对象有:权限[2]、组件[3]、静态 API 序列[4]等.Wu 等[5]提出从 Androidmanifest.xml 文件中提取权限信息作为特征,然后采用 kNN 算法进行分类.Yerima 等[6]用权限和敏感 API 调用来检测恶意软件,实验结果表明这种方法性能尚可.AndroDialysis[7]工具提取应用程序的 intent 作为安卓恶意软件检测的特征.基于签名的检测方法优势在于检测速度快,开销较小,在面对大批量的未知应用时具有较高可用性;劣势在于面对混淆技术、代码动态加载技术和 Java 反射调用时效果不佳.

2.2　基于深度学习的检测

近年来,面对数量骤增的恶意软件,传统的签名检测方式的表现无论从效率还是效果上均已不能满足实际应用的需要,因此研究员们逐渐转向深度学习.深度学习中应用较广的算法如卷积神经网络(CNN)、循环神经网络(RNN)在恶意软件检测中发挥了重要作用[8-10];Nix 等[11]提出基于安卓应用程序的 API 调用对应用进行检测和分类的卷积神经网络;Hou 等[12]提出了一个可以自动检测安卓恶意软件的系统 DroidDelver,不仅从反编译后的 smali 代码中提取 API 调用块的特征,还分析了 API 之间的调用关系,并用深度信念网络(DBN)来检测 Android 恶意软件,结果表明其识别率达到 96.66%,后续可通过增加特征提取的语义深度来进一步改进.Xiao 等[13]提出用恶意软件的系统调用序列训练 LSTM 神经网络,根据模型来预测恶意软件.McLaughlin 等[14]将 APK 反编译得到的 opcode 序列编码,利用 CNN 检测安卓恶意软件.高阳晨等[15]基于 CNN 架构,提取 Android 应用的 DEX 文件然后将其转换成灰度图像,提取图像中的纹理、色彩等信息作为特征并进行训练分类.然而现有的 CNN、RNN 检测模型没能很好地处理安卓程序中函数调用的结构依赖关系,因为对于恶意软件的识别,文本特征固然重要,但其运行逻辑和内部的结构特征也具有很强的可识别性.

2.3　基于图的检测

为从样本中获取更多结构性信息,同时实现比

传统的机器学习和深度学习更高的性能，越来越多的研究人员采用基于图的方法. 如图神经网络(GNN)、图卷积网络(GCN)[16]与图注意力网络(GAT)[17]. Xu 等[18]将安卓应用程序的函数调用图转换为向量，提出了一个基于 GNN 的恶意软件检测系统，该方法再构建模型对恶意软件家族进行分类，其检测准确率达到 99.6%，分类的准确率达到 98.7%. 图卷积网络是一种处理图形数据的半监督方法，它是传统 CNN 的变体，通过局部近似的方式处理频谱图卷积. John 等[19]将安卓应用的系统调用建模为图，并把图的四类中心性度量作为 4 维特征表示输入 GCN，达到了 92.30%的准确率. Oliveira 等[20]提出深度图卷积神经网络(DGCNN)，从应用程序的 API 调用序列中学习特征并进行检测. 赵翠镕等[21]在基于真机的沙箱中运行并提取恶意软件的 API 调用序列，并以关键 API 序列依赖图的典型路径来定义程序行为，最后采用集成学习算法进行恶意代码分类. GAT 是一种新颖的图卷积神经网络. 与 GCN 相比，GAT 用注意力机制代替卷积操作，可以更好地将邻居节点的特征聚合于中心节点. Catal 等[22]针对智能交通系统中的安卓恶意软件提出基于图注意力网络的方法，分别从恶意和良性 APK 文件中提取 API 调用图，虽然取得了比 GCN 更好的效果，但由于使用的是 Node2Vec 嵌入方法，在可解释性上具有一定劣势. Wang 等[23]在 GAT 基础上提出分层的异构注意力网络，通过节点级注意和语义级注意，使该算法具有良好的可解释性. Hei 等[24]将 Android 实体和行为关系建模为异构图，提出一种基于元结构的图注意力网络 MSGAT，并取得了较好的效果. 上述工作表明 GAT 在恶意软件检测领域可以有效地综合图的内容特征和结构特征，达到不错的效果.

3　研究方法

3.1　方法概述

本文方法主要包括 5 个步骤：(1) 利用反编译工具从 APK 中提取 smali 代码；(2) 从 smali 代码中提取 API 调用图，并根据预设的阈值过滤掉边数过少的图；(3) 利用节点嵌入算法，获取每个节点的向量表示；(4) 将向量输入 GAT 网络进行特征学习，将输出后的节点向量累加，形成每个样本对应的图嵌入；(5) 训练分类模型，并在测试集上进行良性和恶意分类实验. 本文所提出方法的流程如图 1 所示.

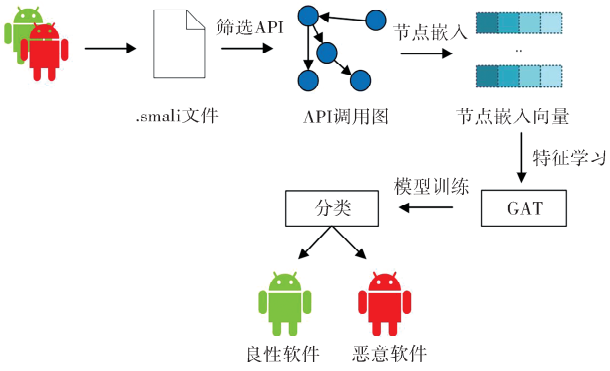

图 1　检测方法流程概览

Fig. 1　Overview of the detection method

3.2　构建 API 调用图

安卓应用程序的核心部分是由 Java 语言编写并编译而成的 Dalvik 虚拟机可执行文件(Dex)，通常以 classes.dex 的形式存在. Smali 代码具有一套独特的指令集，与 Java 中的类一一对应[25]. 方法的第一阶段包含两个步骤.

步骤 1　获取 APK 中的. dex 文件. 使用解压缩工具(如 unzip)将. dex 文件从 APK 压缩文件中解压.

步骤 2　获取 Dalvik 字节码. 使用反编译工具，将 dex 文件反编译为. smali 形式的 Dalvik 字节码. 少部分恶意样本经过反编译后，得到了损坏的 smali 代码，无法进一步生成调用图. 因此将这部分恶意样本排除.

方法的第二阶段是通过 Flowdroid[26]工具构建 FCG. Flowdroid 是一款 Java 编写的安卓静态分析框架，可在不运行程序的情况下分析静态数据流. 本文通过编写 Java 程序，可实现自动从反编译得到的 smali 代码中提取出其中的 API 调用；借助 Java 第三方工具库 GraphViz，将提取出的 API 调用构建为. dot 格式的有向图. 通过研究 API 调用可以有效地判断应用程序的真实目的，例如一个应用程序想要获取设备的 ID，它必须借助安卓系统提供的某个 API 或一系列的 API.

定义 1　函数调用图(FCG)，记为有向图 $G=(V,E)$，其中 $V=\{v_i \mid 1\leqslant i\leqslant n\}$，表示安卓应用程序所调用的函数集合，共调用 n 个函数；$E\subseteq V*V$ 表示函数调用关系的集合，有向边 $\{v_i,v_j\}\in E$ 表示存在从调用者 v_i 到被调用者 v_j 的函数调用关系.

为构建安卓应用程序的 API 调用图，需要定义一个或多个入口点(Entry Point). 大部分安卓应用程序虽然是由 Java 语言编写而成，但两者之

间存在一个很大的不同:Java 程序与 C 语言程序类似,存在一个名为 main 函数的入口点,所有的开发者编写的语句都在 main 函数中才能执行;而安卓应用程序往往存在多个入口点,一些特定的事件触发后才可能执行到入口点中包含的语句,即回调事件. 常见的触发事件有触摸屏幕、解锁屏幕、启动服务等等,而这些事件一般被定义在安卓应用程序的三类组件中,分别是活动(activity)、广播接收器(broadcast receiver)和服务(services),因此在本研究中将这些组件视为提取和构建 FCG 的入口点. 为避免边数过少对本方法性能和准确性的干扰,需要对生成的 API 调用图做进一步处理. 若图中包含的边数过少,则会使节点内容信息的生成中包含的有效信息过少,因此选取有向边的数量满足 $E=\{e_i \mid i \geqslant 50\}$ 的图作为图嵌入阶段的输入.

3.3　SDNE 嵌入

图结构可用于表示各类复杂网络中的特征数据,常见的应用领域有异常检测、节点分类和兴趣推荐等,然而直接分析图结构的时间和空间开销都非常大. 为了让图的丰富信息得以保留,同时减少时间和空间损耗,图嵌入方法应运而生. 图嵌入方法可以将无向图、有向图等各类图或转换为低维的特征向量,或将图的一部分(如部分节点、边或子图)表示为一组包含了图中隐藏信息的向量,再将向量输入各类机器学习或深度学习网络中参与复杂的检测和分类任务. Node2Vec[27] 是基于 DeepWalk[28] 的随机游走策略的改进方法,尽管其结果远好于 DeepWalk,但并未考虑节点自身具有的属性,同时需要手动调节 p、q 值以控制其更偏向于 BFS 或 DFS,参数的具体数值需要凭借经验调节,在可解释性上具有劣势;Pektas 等[29] 使用 API 调用图来表示恶意程序在其运行期间所有可能的执行路径. API 调用图被嵌入到一个低维度的数字向量特征集中,并被引入到深度神经网络中. 作者评估了 DeepWalk、Node2vec、SDNE 和 HOPE 四种不同的图形嵌入方法,发现 SDNE 提供了更多的鉴别特征,并且达到很高的准确率. 因此本文选取 SDNE 算法作为节点嵌入方法.

SDNE(Structural Deep Network Embedding)[30] 是一种基于深度自编码器的图嵌入算法,由两个模块组成:无监督部分的深度自编码器和监督部分的拉普拉斯特征映射. 在使用 SDNE 之前,需要对图中信息进行编码,生成图对应的邻接矩阵.

定义 2　一阶相似度. 记图中任意一对节点 v_i,v_j 的一阶相似度为 $s_{i,j}$,若 $s_{i,j}>0$,则称节点 v_i 与 v_j 之间存在正一阶近似;否则,v_i 与 v_j 之间的一阶相似度为 0.

定义 3　二阶相似度. 记 $N_u=\{s_{u,1},\cdots,s_{u,v}\}$ 为节点 v_u 与其它节点之间的一阶相似度,节点 v 与 u 之间的二阶相似度由 N_u 与 N_v 之间的相似度决定.

从上述两个定义可知,一阶相似度更偏重描述指定的两个节点之间的相似性,而二阶相似性更偏重描述指定的两个节点的相邻结构之间的相似性. 以上公式成立基于二阶相似性假设,即:如果两个节点有许多共同的邻居节点,那么就认为它们在很大程度上是相似的. SDNE 的拉普拉斯特征映射模块负责计算给定图中节点的一阶相似度,若节点 v_i 和 v_j 之间存在一条边,则认为嵌入后的映射结果相似. 一阶相似度的损失函数如式(1)所示.

$$L_1=\sum_{i,j=1}^{n} s_{i,j} \parallel y_i^{(K)}-y_j^{(K)} \parallel_2^2 = \sum_{i,j=1}^{n} s_{i,j} \parallel y_i-y_j \parallel_2^2 \tag{1}$$

自编码器模块负责处理二阶相似度,接收输入为节点 i 的邻接矩阵 x_i,由于每一个邻接矩阵中都包含了节点 i 的邻居结构信息,因此结构相似的节点会学习到相似的嵌入向量,然后通过优化损失函数确定二阶相似度. 但由于图结构本身具有稀疏性,节点的邻接矩阵中非零元素远少于零元素,这样会造成即使输入全零矩阵也能获得一定的效果,这就降低了矩阵中非零元素的重要性. 为避免此种情况,需要优化损失函数,优化后的损失函数如式(2)所示.

$$L_2=\sum_{i}^{n} \parallel (\hat{x_i}-x_i)\odot b_i \parallel_2^2 = \parallel (\hat{X_i}-X_i)\odot B \parallel_F^2 \tag{2}$$

其中,$\odot$ 是哈马达乘积,表示对应元素逐个相乘;$b_i=\{b_{i,j}\}_{j=1}^{n}$,若 $s_{i,j}=0$,则 $b_{i,j}=1$,否则 $b_{i,j}=\beta>1$. 由此实现为有边连接的节点赋予更高权重. 对于一张节点数为 N 的 API 调用图,最终可生成 N 个 F 维向量,以参与后续图注意力网络的训练.

3.4　图注意力网络模型

上述过程已经为每个样本所生成的 API 调用图都生成了节点嵌入向量组,图注意力模型采用注意力机制替代了 GCN 中的卷积核运算. 假设一张图 $G=(V,E)$ 有 N 个节点,输入图注意力网络的每个节点的 F 维特征表示为 $h=\{\vec{h}_1,\vec{h}_2,\ldots,\vec{h}_N\}$,$\vec{h}_i\in\mathbb{R}^F$,输出的所有节点特征表

示为$h'=\{\vec{h}_1',\vec{h}_2',\ldots,\vec{h}_N'\},\vec{h}_i'\in\mathbb{R}^{F'}$. 在此过程中需要一个可学习的线性变换 W 实现特征增强，将输入特征转化为高阶特征，因此节点特征向量的维度会发生变化，即 $F\to F'$. 设节点v_i和v_j的特征为$\vec{h}_i$，$\vec{h}_j$，需要对所有节点训练一个权值矩阵 $W\in\mathbb{R}^{F'\times F}$，从 F 维转化为F'维新特征$\vec{h}_i'$，$\vec{h}_j'$. 对于节点v_i，式(3)计算其所有邻居节点v_j与它之间的注意力互相关系数e_{ij}.

$$e_{ij}=a([W\vec{h}_i\,\|\,W\vec{h}_j]),j\in N_i \tag{3}$$

其中，a 表示一个共享注意力机制：$\mathbb{R}^{F'}\times\mathbb{R}^{F'}\to\mathbb{R}$，可通过一个单层的前馈神经网络实现将高维特征映射到实数空间. 需要说明的是，此公式是具有方向性的，表示节点v_j对于v_i的重要性. 为使得互相关系数便于比较，式(4)引入非线性激活函数 LeakyReLU 对v_i的所有相邻节点v_j进行归一化.

$$\alpha_{ij}=\frac{\exp(LeakyReLU(e_{ij}))}{\sum_{k\in N_i}\exp(LeakyReLU(e_{ik}))} \tag{4}$$

根据上述过程计算得出的注意力系数，再对特征进行加权求和，最后通过一个具有非线性特征的激活函数 σ 进行正则化得到每个节点的输出. 如式(5)所示.

$$\vec{h}_i'=\sigma\Big(\sum_{j\in N_i}\alpha_{ij}W\vec{h}_j\Big) \tag{5}$$

本文提出的安卓恶意软件检测的图注意力网络模型由两层图注意力层和一层全连接层组成，其中前两层为特征学习层，第三层为分类层. 将由 SDNE 生成的图嵌入向量输入网络中的注意力层，两层学习层会根据每张图的所有节点对应的图嵌入向量进行注意力学习得到输出节点向量，再将每张图的所有输出节点向量进行求和，形成图嵌入；分类层采用全连接方式，输出为良性－恶意互斥的二分类，激活函数采用 sigmoid. 图注意力网络结构如图 2 所示.

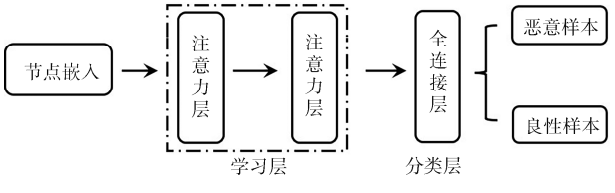

图 2　图注意力网络结构

Fig. 2　Structure of graph attention networks

4　实验评估

4.1　实验环境与数据集

本文中恶意软件样本均来自 AMD 数据集[31]. AMD 是由 ArgusLab 于 2017 年公开共享的安卓恶意软件数据集，包含 2010 至 2016 年获取的 24 650 个恶意样本，分为 135 类，共 71 个恶意软件家族. 随机选取 AMD 数据集中的 1200 个恶意样本，600 个 Google 应用商店和 600 个 360 应用商店的良性样本，并对每个样本都进行了 2 种常见杀毒软件的检查，以确保良性和恶意样本的有效性. 通过检查后将它们作为数据集，分别取恶意和良性各 200 个作为测试集.

硬件环境采用 Intel Xeon CPU E3-1231 v3 处理器，32 GB 内存，NVIDIA GTX 1050Ti GPU，软件环境采用 Ubuntu 20.04 LTS，Pytorch-1.8.1 工具库.

表 1　数据集样本分布

Tab. 1　Database distribution

样本类别	训练样本集	测试样本集
恶意样本	1000	200
良性样本	1000	200
合计	2000	400

4.2　实验评估指标

为评估检测效果，本文使用四种性能度量：精度(Precision)、召回率(Recall)、F_1 分数(F-Score)和准确率(Accuracy)作为评估指标，这些性能度量均通过真阳性(TP)、假阳性(FP)、真阴性(TN)和假阴性(FN)四个检验指标运算得出. 本文中性能度量的数学定义如下.

1) 精度：正确判定的恶意样本数量与所有判定为恶意样本的数量之比.

$$Preciosion=\frac{TP}{TP+FP}$$

2) 召回率：正确判定的恶意样本数量与全部真正的恶意样本数量之比.

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

3) F_1分数：精度和召回率的加权调和平均值.

$$F\text{-}Score=\frac{2*Precision*Recall}{Precision+Recall}$$

4) 准确率：正确判定良性和恶意样本的数量与所有样本数量之比，简写为 Acc.

$$Acc=\frac{TP+TN}{TP+TN+FP+FN}$$

在上述指标中，精度和召回率的计算公式分子相同，但分母中的 FP 与 FN 是相互矛盾的，因此无法同时达到很高的水平. F_1分数同时考察精度

和召回率,可在二者之间达到平衡,其值越高表明模型越好.

4.3　图嵌入长度评估实验

图嵌入算法本质是将图信息映射到向量空间,在此过程中嵌入长度会对特征向量的生成有一定影响.为找出最适合本方法的嵌入长度,进行图嵌入长度评估实验.一方面,在 3.2 节中只选取了节点数不少于 50 的图,因此嵌入长度应需要尽量不高于 50 才能覆盖所有样本生成的图;另一方面,若嵌入长度大于 32,可能会造成向量维度过高,反而失去图嵌入方法的优势.为有效生成图嵌入向量,实验选择从 8 开始,以 8 为步长,最长为 32 进行评估,得到不同嵌入长度时的检测结果如表 2.

表 2　不同嵌入长度下的检测结果

Tab. 2　The results of different embedding size

嵌入长度	准确率/%	精度/%	召回率/%	F_1分数/%
8	95.97	96.24	95.89	96.26
16	96.61	96.65	96.21	96.43
24	97.09	96.70	96.44	96.57
32	97.23	96.74	96.49	96.59

由表 2 数据得知,嵌入长度从 8 增加至 16 的过程中,准确率增加了 0.64%,F_1 分数增加了 0.17%,但从 16 开始逐步增加到 32 的过程中,这两项指标的增加幅度开始下降,分别为 0.48%、0.14%和 0.14%、0.02%.这组数据变化的幅度表明,尽管提升嵌入长度的能够使检测结果更加准确,但嵌入长度增加所带来的开销增也很大,为了均衡检测效果和开销,选择嵌入长度为 24.

4.4　Dropout 率评估实验

Dropout 是一种防止神经网络在训练时出现过拟合现象的方法[32],其原理是从神经网络中随机排除一些单元.此实验旨在通过对比不同 Dropout 率时所训练网络的检测效果确定最适合本方法的 Dropout 率,并在后续与其他研究者所提出的方法对比时保持恒定.

由表 3 数据可知,当 Dropout 为 0 时检测效果不够好,以 10%为间隔逐步增加 Dropout 率,当增加到 30%时模型取得了最好的效果;当 Dropout 率从 30%增加到 50%的过程中检测效果有所下降.这表明较低的 Dropout 率有助于模型的学习,尽可能地避免了过拟合的现象,而过高的 Dropout 率则会使模型无法充分地学习,因此选择 Dropout 率为 0.3 为最佳.

表 3　不同 Dropout 率的检测结果

Tab. 3　The results of different dropout rate

Dropout 率	准确率/%	精度/%	召回率/%	F_1分数/%
0	95.51	94.57	95.64	95.10
0.1	96.32	95.68	95.23	95.45
0.2	96.39	97.05	96.54	96.79
0.3	97.87	97.34	97.47	97.40
0.4	97.72	97.12	97.43	97.27
0.5	94.41	94.80	92.68	93.73

4.5　与同类工作对比实验

上述两个实验表明,本文所提出的方法在嵌入长度调整为 24,Dropout 设置为 0.3 时效果最佳.本小节将本文方法与三个相关工作进行对比.Xiao 等[13]利用动态分析,提取恶意软件的系统调用序列训练 LSTM 神经网络,以预测恶意软件;McLaughlin 等[14]将 APK 反编译得到的 opcode 序列进行编码,随后利用卷积神经网络进行检测;Catal 等[22]从恶意和良性 APK 文件中提取 API 调用图,并使用 Node2vec 生成嵌入后输入 GCN 和 GAT 中进行检测.表 4 总结了四种方法在不同数据集上的准确率、精度、召回率和 F_1 分数,可以看出本方法所选取的 Android 程序在转换为调用图时保留了程序的结构特征,同时后续的图注意力网络将节点之间的相互影响程度进行了计算和加权,使得生成的图嵌入向量包含了丰富的、可以区分重要性的邻居特征.本文提出的模型在各项指标中均优于其他三种方法,在安卓恶意软件检测方面性能更佳.

表 4　与同类工作结果比较

Tab. 4　Comparison with existing works

检测方法	准确率/%	精度/%	召回率/%	F_1分数/%
Xiao[13]	93.10	94.76	91.31	93.00
McLaughlin[14]	96.30	97.27	97.25	97.26
Catal[22]	96.10	95.21	94.39	94.80
本文方法	97.87	97.34	97.47	97.40

5　结　论

本文提出了一种基于图注意力网络的安卓恶意软件检测方法.该方法首先对安卓应用进行静态分析,构建 API 调用图以初步表征 APK;其次引入 SDNE 算法为每个节点生成嵌入;随后将节点

嵌入输入图注意力网络形成图嵌入；最后进行检测任务. 为使模型达到最优效果，在正式实验之前分别进行了嵌入长度和 Dropout 率调优. 实验结果表明，与现有的一些方法相比，本文的方法达到了最好的准确率和 F_1 分数，分别为 97.87% 和 97.40%，因此本方法在安卓恶意软件检测方面效果更佳. 在未来的工作中，将尝试采用混合分析的方法，把恶意软件的行为特征也纳入考虑，并探索能更好表达图中节点特征的图嵌入算法，以取得更好的检测效果.

参考文献：

[1] 360 互联网安全中心. 2021 年度中国手机安全状况报告[EB/OL]. (2022-01-25)[2022-02-08]. https://pop.shouji.360.cn/safe_report/Mobile-Security-Report-202112.pdf.

[2] Wang W, Wang X, Feng D, *et al*. Exploring permission-induced risk in Android applications for malicious application detection [J]. IEEE T Inf Foren Sec, 2014, 9: 1869.

[3] Saracino A, Sgandurra D, Dini G, *et al*. Madam: effective and efficient behavior-based Android malware detection and prevention [J]. IEEE T Depend Secure, 2016, 15: 83.

[4] Zhu J, Wu Z, Guan Z, *et al*. API sequences based malware detection for Android [C]//Proceedings of the 2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom). Beijing: IEEE, 2015.

[5] Wu D J, Mao C H, Wei T E, *et al*. Droidmat: Android malware detection through manifest and api calls tracing [C]//Proceedings of the 2012 Seventh Asia Joint Conference on Information Security. Tokyo, Japan: IEEE, 2012.

[6] Yerima S Y, Sezer S, Muttik I. High accuracy android malware detection using ensemble learning [J]. IET Inform Secur, 2015, 9: 313.

[7] Feizollah A, Anuar N B, Salleh R, *et al*. AndroDialysis: Analysis of android intent effectiveness in malware detection [J]. Comput Secur, 2017, 65:121.

[8] Aafer Y, Du W, Yi H. Droid API miner: mining api-level features for robust malware detection in android [C]//Proceedings of the 9th International Conference on Security and Privacy in Communication Networks. Sydney: Springer, 2013.

[9] Yuan B, Wang J, Liu D, *et al*. Byte-level malware classification based on markov images and deep learning [J]. Comput Secur, 2020, 92: 101740.

[10] Huang N, Xu M, Zheng N, *et al*. Deep android malware classification with api-based feature graph [C]//Proceedings of the 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). Rotorua: IEEE, 2019.

[11] Nix R, Zhang J. Classification of Android apps and malware using deep neural networks [C]//2017 International joint conference on neural networks (IJCNN). Anchorage: IEEE, 2017.

[12] Hou S, Saas A, Ye Y, *et al*. DroidDelver: an android malware detection system using deep belief network based on API call blocks [C]// Proceedings of the Web-Age Information Management. Lecture Notes in Computer Science. Cham: Springer, 2016.

[13] Xiao X, Zhang S, Mercaldo F, *et al*. Android malware detection based on system call sequences and LSTM [J]. Multimed Tools Appl, 2019, 78: 3979.

[14] Mclaughlin N, Doupé A, Ahn G J, *et al*. Deep android malware detection [C]// Proceedings of the ACM on Conference on Data and Application Security and Privacy. Scottsdale: ACM, 2017.

[15] 高杨晨，方勇，刘亮，等. 基于卷积神经网络的 Android 恶意软件检测技术研究[J]. 四川大学学报：自然科学版，2020，57：673.

[16] Welling M, Kipf T N. Semi-supervised classification with graph convolutional networks [C]. [S.l.]: ICLR, 2016.

[17] Veličković P, Cucurull G, Casanova A, *et al*. Graph attention networks [EB/OL]. (2018-02-04)[2022-02-08]. https://arxiv.org/pdf/1710.10903.pdf.

[18] Xu P, Eckert C, Zarras A. Detecting and categorizing Android malware with graph neural networks [C]//Proceedings of the 36th Annual ACM Symposium on Applied Computing. Korea: ACM, 2021.

[19] John T S, Thomas T, Emmanuel S. Graph convolutional networks for android malware detection with system call graphs [C]//Proceedings of the 2020 Third ISEA Conference on Security and Privacy. Guwahati: IEEE, 2020.

[20] Oliveira A, Sassi R. Behavioral malware detection using deep graph convolutional neural networks [J]. Int J Comput Appl, 2019, 975: 8887.

[21] 赵翠镕, 方勇, 刘亮, 等. 基于语义 API 依赖图的恶意代码检测[J]. 四川大学学报: 自然科学版, 2020, 57: 488.

[22] Catal C, Gunduz H, Ozcan A. Malware detection based on graph attention networks for intelligent transportation systems [J]. Electronics, 2021, 10: 2534.

[23] Wang X, Ji H, Shi C, *et al*. Heterogeneous graph attention network [C]// Proceedings of the World Wide Web Conference. [S. l.]: ACM, 2019.

[24] Hei Y, Yang R, Peng H, *et al*. Hawk: Rapid android malware detection through heterogeneous graph attention networks [J]. IEEE T Neur Net Lear, 2021(99): 1.

[25] 范铭, 刘烃, 刘均, 等. 安卓恶意软件检测方法综述[J]. 中国科学: 信息科学, 2020, 50: 1148.

[26] Arzt S. FlowDroid [EB/OL]. [2022-01-08]. https://github.com/secure-software-engineering/FlowDroid.

[27] 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.

[28] Perozzi B, Al-Rfou 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.

[29] Pektaş A, Acarman T. Deep learning for effective Android malware detection using API call graph embeddings [J]. Soft Comput, 2020, 24: 1027.

[30] 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.

[31] Wei F, Li Y, Roy S, *et al*. Deep ground truth analysis of current android malware [C]// Proceedings of the 2017 Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA. Lecture Notes in Computer Science. Cham: Springer, 2017.

[32] Srivastava N, Hinton G, Krizhevsky A, *et al*. Dropout: a simple way to prevent neural networks from overfitting [J]. J Mach Learn Res, 2014, 15: 1929.

引用本文格式:

中　文: 岳子巍, 方勇, 张磊. 基于图注意力网络的安卓恶意软件检测[J]. 四川大学学报: 自然科学版, 2022, 59: 053002.

英　文: Yue Z W, Fang Y, Zhang L. Android malware detection based on graph attention networks [J]. J Sichuan Univ: Nat Sci Ed, 2022, 59: 053002.