基于深度学习的Android恶意软件检测：成果与挑战

陈 怡[①③④] 唐 迪[②] 邹 维*[①③④]

[①](中国科学院信息工程研究所 北京 100093)

[②](香港中文大学 香港 999077)

[③](中国科学院网络测评技术重点实验室 北京 100093)

[④](中国科学院大学网络空间安全学院 北京 100049)

摘 要：随着Android应用的广泛使用，Android恶意软件数量迅速增长，对用户的财产、隐私等造成的安全威胁越来越严重。近年来基于深度学习的Android恶意软件检测成为了当前安全领域的研究热点。该文分别从数据采集、应用特征、网络结构、效果检测4个方面，对该研究方向已有的学术成果进行了分析与总结，讨论了它们的局限性与所面临的挑战，并就该方向未来的研究重点进行了展望。

关键词：移动安全；Android恶意软件；Android应用；深度学习；机器学习

中图分类号：TP309.5 **文献标识码**：A **文章编号**：1009-5896(2020)09-2082-13

DOI: 10.11999/JEIT200009

Android Malware Detection Based on Deep Learning: Achievements and Challenges

CHEN Yi[①③④] TANG Di[②] ZOU Wei[①③④]

[①](*Institute of Information Engineering, Chinese Academy of Sciences, Beijing* 100093, *China*)

[②](*Chinese University of Hong Kong, Hongkong* 999077, *China*)

[③](*Key Laboratory of Network Assessment Technology, Chinese Academy of Sciences, Beijing* 100093, *China*)

[④](*School of Cyber Security, University of Chinese Academy of Sciences, Beijing* 100049, *China*)

Abstract: With the prosperous of Android applications, Android malware has been scattered everywhere, which raises the serious security risk to users. On the other hand, the rapid developing of deep learning fires the combat between the two sides of malware detection. Inducing deep learning technologies into Android malware detection becomes the hottest topic of society. This paper summarizes the existing achievements of malware detection from four aspects: Data collection, feature construction, network structure and detection performance. Finally, the current limitations and facing challenges followed by the future researches are discussed.

Key words: Mobile security; Android malware; Android application; Deep learning; Machine learning

1 引言

Android系统是由Google公司于2007年11月正式发布的一款面向智能移动设备的开源操作系统，其开放自由的特性使得Android系统迅速普及。根据最新统计，截止至2019年10月，Android系统仍是市场占有率最大的智能移动设备操作系统，其占有率高达87%[1]。与此同时，大量的Android恶意软件被发布到Android应用(Application, App)市场上。据腾讯安全移动实验室报告，在2019年上半年，共发现1898731款新的Android恶意软件，它们感染了约3813万的手机用户[2]。这些恶意软件破坏用户系统、窃取用户隐私、恶意扣除资费等，对用户的信息安全和财产安全造成了严重的威胁。因此，检测Android恶意软件的需求十分迫切，是当前信息安全领域的重要任务之一。

针对Android恶意软件检测这个课题，早期研究多使用基于指纹的检测方法。该类方法的检测准确率较高，但是它们的检测效果依赖于指纹的质量和数目。而这些指纹不仅往往需要安全研究人员耗费人力手工提取，同时恶意软件通过简单的变化即可轻易改变其指纹，进而绕过这类安全检测方法[3]。

收稿日期：2020-01-20；改回日期：2020-07-30；网络出版：2020-08-07

*通信作者： 邹维 zouwei@iie.ac.cn

基金项目：中国科学院重点实验室基金(CXJJ-19S022)

Foundation Item: Foundation of Key Laboratory of Network Assessment Technology, Chinese Academy of Sciences (CXJJ-19S022)

基于机器学习的Android恶意软件检测方法可以弥补以上缺点。它不依赖于某个固定的指纹，而是通过机器学习算法，依照事先设计好的目标函数，自动地提取出最合适的特征及特征的组合来判断Android应用是否为恶意软件。在机器学习算法中，随着硬件计算能力的提升与数据的大量累积，相较于决策树(decision tree)[4]、支持向量机(support vector machine)[5]、朴素贝叶斯(naive Bayes)[6]等传统的机器学习算法，深度学习已成为现阶段最热门的算法。它是目前公认的对分类检测问题十分有效的算法，在图像识别、音频分析、自然语言处理等领域已取得了丰硕的研究成果[7–9]。近年来，研究人员在Android恶意软件检测领域也累积了海量的数据，这为深度学习在该领域的发展提供了数据支撑。因此，如何将深度学习应用到Android恶意软件检测中来提高恶意软件检测效果成为了近年来该领域的研究热点。

现有的基于深度学习的Android恶意软件检测工作都将深度学习算法作为一个分类器使用，把Android应用分为良性软件和恶意软件这两类。这些工作一般含有4个主要步骤：(1)数据采集；(2)特征提取；(3)模型设计；(4)效果检测。

(1) 数据采集：数据采集在深度学习中是关键的第1步，采集一个充分的数据集可以有效提高在其上训练出来的模型的分类准确率和范化效果。因此本文在第2节梳理了现有工作在研究当中所常用的数据集及数据采集的方法。

(2) 特征提取：尽管源数据可以直接作为深度学习模型的输入，但是当源数据过于庞大时(如一个App的大小可能高达数GB)，现阶段深度学习模型还不能从中很好地提取出Android应用的程序特性。为了进一步加强模型对Android应用的理解，提高深度学习模型的检测效果，大部分现有工作采用的方法是从源数据中提取出一些描述Android应用的特征，将这些特征作为深度学习模型的输入数据。本文在第3节归纳并整理了现有工作提取的特征类型及特征构造的方法。

(3) 模型训练：模型训练这一步是在已经构造好的模型输入数据集上，选择合适的模型进行训练的过程。在已有的深度学习的研究当中出现了许多各具特色的网络结构。本文在第4节归纳总结了在Android恶意软件检测中常用的深度学习网络结构，并讨论其各自的特点。

(4) 效果检测：在获得深度学习模型后，从一个严谨的检测机制中可以获得较为客观准确的模型效果评价。因此本文在第5节归纳总结并对比了已有研究所提出的方案的检测效果，以及它们在检测时存在的缺点。

本文依照上述这4个步骤，分别从数据采集、应用特征、网络结构、效果检测这4个方面，对基于深度学习的Android恶意软件检测研究方向现有的学术成果进行了分析与总结，讨论了它们的局限性与所面临的挑战，并提出了可能的解决思路，最后对该方向未来的研究重点进行了展望。

2　数据采集

在深度学习中，训练集数据量的大小直接影响着模型的效果：如若在小数据集上进行深度学习则易出现过拟合现象。因此，采集一个均衡的、充分的数据集用于训练十分重要。现有工作所使用的数据主要有3个来源：应用市场、公共数据集、安全公司内部数据。下文将分别针对这3个来源进行介绍。

2.1　应用市场

随着Android系统的广泛使用，Android应用的种类、数量都不断攀升。截止至2019年11月，Android官方应用市场Google Play可用App高达290万例[10]。此外，许多第三方应用市场也发布了大量的Android应用，譬如国内当前热门的百度手机助手[1)]、小米应用商店[2)]、华为应用市场[3)]等，它们都是手机用户下载Android应用的主要来源。因此，许多研究工作[11–23]扒取了Google Play或第三方应用市场的Android应用作为其数据来源。

但是，由于Android生态系统的开放性，市场中存在着许多恶意软件。为了标记它们，部分工作[14,17,21,23]利用了VirusTotal[4)]来鉴别Android应用的好坏。VirusTotal是一个免费的针对病毒、蠕虫、木马等各种恶意软件的分析服务，能够快速扫描发现Android恶意软件。VirusTotal集成了70个杀毒软件，包括国内外著名的杀毒软件McAFee、赛门铁克、卡巴斯基、360杀毒等。对于上传到VirusTotal服务器进行分析的Android应用，VirusTotal会迅速返回70个杀毒软件对该应用的检测报告。利用VirusTotal的服务，可实现对从Android应用市场扒取的大量App的自动标定，形成深度学习的训练数据集。

2.2　公共数据集

许多已有的研究工作将他们在研究时所收集的

1)百度手机助手：https://shouji.baidu.com
2)小米应用商店：http://app.mi.com
3)华为应用市场：https://appstore.huawei.com
4)VirusTotal：https://www.virustotal.com

Android应用数据集进行了开放，具有代表性的有文献[24–30]。其中文献[24–27]的主要成果是收集了大量的Android恶意软件供研究人员对Android应用进行研究，而文献[28–30]在研究Android恶意软件的同时，也公开了它们所使用的数据集。本文对这些数据集进行了总结，从恶意软件数量、软件收集时间、软件检测方法3个方面进行了归纳，并给出了数据集的下载链接，详见表1。

表 1 Android恶意软件公开数据集统计表

数据集名称	恶意软件数量	软件收集时间	软件检测方法	下载链接
VirusShare[24]	34311879	2011至今	未说明	https://virusshare.com
AndroZoo[25]	1302968	2011至今	VirusTotal	https://androzoo.uni.lu
ArgusLab[26]	24650	2010～2016	VirusTotal	http://amd.arguslab.org
Drebin[28]	5560	2010～2012	VirusTotal	http://contagiominidump.blogspot.com
ISCX[29]	1929	2012～2015	VirusTotal	https://www.unb.ca/cic/datasets/index.html
Genome[30]	1260	2010～2011	未说明	http://www.malgenomeproject.org
Contagio[27]	252	2011～2018	未说明	http://contagiominidump.blogspot.com

从表1中可以看到，这些数据集涵盖了2010年至今不同时期的Android恶意软件。尽管它们由不同组织各自收集，可能存在部分交集。但是，从VirusShare的数据集中可以知道，目前已知的Android恶意软件数量高达数千万，这些数量庞大的数据集为有效的深度学习模型训练提供了基础。

2.3 安全公司数据

除以上讨论的从Android应用市场与公开数据集获得数据外，部分已有工作[20,31–35]通过与安全公司的合作获得公司内部的Android应用数据集用于研究。表2对它们合作的公司及获得的数据量进行了整理。其中，文献[35]在获取公司数据后，还将研究中所使用的72GB实验数据进行了公开[5)]，供该领域其他学者使用。

2.4 小结与讨论

通过梳理现有工作所使用的数据集可以看到，无论是Android良性软件还是Android恶意软件，其数据量均达到了千万级别，如此庞大的数据量可有效支撑深度学习的模型训练。但在数据采集过程中，部分工作[11–23]直接把从应用市场上爬取的所有应用都标记为良性软件作为训练集使用是不合理的。众所周知，Android应用市场上存在着大量的Android恶意软件，如此标定的数据集与良性软件和恶意软件的实际数据集分布不同，所训练出的模型在真实场景下容易产生漏报，即误将恶意软件判断为良性软件。此外，当前研究对训练集的标定总体上都依据于VirusTotal杀毒软件的报告结果。工业界的杀毒软件为了避免误报，往往会采用特征相对固定的基于指纹的恶意软件检测方法。当深度学习在这些方法标定的数据集上去学习特征时，其在实验室环境下测试所得的模型检测准确率往往会偏高，但对于未知恶意代码的检测准确率很可能会偏低。因此，在未来的研究工作中，可以就训练集中的恶意软件进行适当地变化，利用代码混淆、加密等手段来丰富训练集的多样性，提高模型在现实中的恶意代码检测的准确率。

表 2 公司合作及数据采集统计表

文献	合作公司	良性软件	恶意软件
文献[31]	腾讯安全实验室	83784	106912
文献[32]	McAfee	19620	11505
文献[20]	McAfee	3627	2475
文献[33]	Comodo	2500	2500
文献[34]	Comodo	1500	1500
文献[35]	Leopard Mobile Inc	2000000	

[5)]下载地址：http://R2D2.TWMAN.ORG

3 应用特征

尽管深度学习具有从源数据中自动学习特征的能力，但是当源数据过于庞大时，现阶段的深度学习还不能从中很好地提取出Android应用的程序特性。因此，为了进一步加强模型对Android应用的理解，提高深度学习模型的检测效果，绝大部分的工作没有直接使用源数据，而是从Android应用中提取出描述应用特性的特征，将它们作为深度学习模型的输入数据。针对这一重要过程，下文对这些工作所提取的特征类型和特征构造的方法进行了总结与讨论。

3.1 特征类型

现有工作所使用的特征主要可以分为3类：基于元数据的特征类型、基于指纹的特征类型和基于行为的特征类型。下文分别对这3种应用特征类型进行介绍。

3.1.1 基于元数据的特征类型

元数据是描述Android应用的数据，如Android应用安装包的大小、Android应用程序中组件的数量等。同时，元数据还表示Android应用本身的一些属性，如应用的签名、软件开发工具版本、应用运行时的硬件要求等。这些数据虽无法直接说明一个Android应用是否为恶意软件，但是它们从侧面反应了应用的能力及可能的行为。因此，文献[11,17,21,22,31,36,37]均使用了Android应用的元数据作为应用特征。

3.1.2 基于指纹的特征类型

Android恶意软件的指纹能够以极高的准确率检测出恶意软件，但指纹的提取往往需要耗费大量安全人员的人力。深度学习具有从输入数据中筛选特征并构建相关特征与最终分类关联性的能力，因此部分工作提取出Android应用中可能的指纹字段作为应用特征输入到模型中，让深度学习自动学出其中实际能帮助检测恶意软件的指纹、指纹组合及组合方式。

常见的恶意软件指纹包括程序所使用的特殊字符串、远程链接的恶意服务器网址，以及编程所使用的特殊函数名、变量名等。基于此，文献[11]提取出Android应用源代码中所使用的全部字符串常量，文献[36,37]提取出Android应用源代码中出现的所有的URL地址，文献[17,32,36,37]提取出Android应用源代码中所有组件的函数名、Intent的命名。这些文献将它们作为深度学习模型的输入，让深度学习在训练的过程中自动在这些备选项中找到最能区分恶意软件的特征组合来检测Android恶意软件。

3.1.3 基于行为的特征类型

Android恶意软件的本质在于其会执行一些恶意行为，因此许多工作提取了能描述应用行为的特征作为深度学习模型的输入。本文将它们所提取的行为细分为3类：(1)运行时的行为；(2)代码编写的行为；(3)具备的行为能力。

(1) 运行时的行为：系统调用(system call)是软件运行时操作系统的底层接口，它能较为准确地反映软件的实际行为。文献[13,16,18,33]均利用系统调用设计了应用特征。具体来说，文献[18]将Android应用运行时所产生的系统调用序列直接作为特征输入到循环神经网络中；文献[33]认为恶意软件可能会频繁地使用某些系统调用，因此将系统调用的频数作为描述应用行为的特征；文献[13,16,33]观察到恶意行为通常是由多个系统调用配合完成，为了描述这些配合，文献[13,33]计算了任意两个系统调用在Android应用运行时连续出现的概率，文献[33]还计算了任意两个系统调用在系统调用序列里平均的位置距离，将它们作为描述Android应用运行时的行为的特征。

(2) 代码编写的行为：Google向Android应用开发者提供了大量的接口(APplication Interface, API)来操作、使用移动设备的功能，如发送短信、拍照摄像、读通讯录等。因此，应用代码中Android API的调用可以反映开发者期望代码去实现的行为。文献[11,12,15,17,19,23,32,34,36,37]均使用了Android API的相关信息来设计应用特征。同上述系统调用的特征提取方法类似，文献[12,15,17,19,23,32,34,36,37]将API的使用与否作为应用特征。文献[11,15]则从恶意行为需要多个API配合的观点出发来提取特征，文献[15]检查了任意两个API的配合关系，将两个API是否在同一个函数出现、是否在相邻的函数出现作为特征；文献[11]则计算了整个应用的API序列与恶意软件库中API序列的相似性，将它们的欧氏距离作为特征。由于Android API数量约为3万个，如果提取所有Android API的相关特征作为深度学习模型的输入，这样的模型将会需要大量的数据进行训练。为此，这些工作都只关注了与恶意行为可能相关的敏感API，如文献[36]只关注了文献[38]所给出的恶意软件相关敏感API列表中的API。

(3) 具备的行为能力：在Android系统中权限保护机制是Android应用的重要安全机制之一。当应用要执行某些敏感操作时，开发者需要在软件开发时在AndroidManifest.xml文件中声明对应的保护权限。这些权限的声明意味着应用具备了执行其对应的敏感操作的能力。文献[11,17,19,22,32,36,37]均将Android应用中的权限作为应用特征来描述应用可能的行为。其中，文献[11,17,19,22,32]不仅检查了应用在AndroidManifest.xml中所声明的权限，同时还进一步扫描了软件代码来提出代码编写过程中实际使用到的权限。

3.2 特征构造

在上节中本文讨论了应用特征的类型，但很多类型的特征都无法直接作为深度学习模型的输入，比如系统调用、特殊字符串、权限等。特征构造便是首先将它们量化为数值，然后再构造成深度学习模型所能接受的数据结构。因此，下文对现有工作特征量化与模型输入构造的方法分别进行了归纳总结。

3.2.1 特征量化

现有工作特征量化主要可以分为两类：(1)基于存在性的特征量化；(2)基于转换的特征量化。

(1) 基于存在性的特征量化：当特征数量有限

时，许多工作会基于特征的存在性进行量化，具体来说主要包括两种方法：将所有特征进行编号和使用0-1向量表示特征存在与否。例如，文献[14,18,34,36]对Android API这个特征进行了编号，用API编号形成序列、向量或矩阵输入到深度学习模型中；文献[11,15,17,19,21,23,32,37]则使用了第2种方法，当某个特征从Android应用中成功提取出时，其对应的位置置为1，否则为0。

(2) 基于转换的特征量化：文献[31,39]的特征均为图。其中文献[31]构造了一个异构图(Heterogeneous Graph, HG)[40]，把每个应用和应用的特性(如签名、所使用的API等)均作为图中的节点，后者与对应的应用节点相连；然后，基于异构图镶嵌(HG embedding)的方法[41–43]将图中的每个节点转换为一个向量；通过这样的方法，使用应用节点的向量来作为深度学习模型的输入。文献[39]提出了应用的API的调用图，同样文献[39]也使用了一个基于异构图镶嵌的方法将整个图转换为一个矩阵来表示，作为深度学习模型的输入。此外，对于深度学习模型的输入数据来说，一个大的取值范围会增加模型的拟合难度，故而将输入数据的取值范围进行压缩会使得训练更加容易。因此，文献[11]将表示API序列与恶意软件API序列相似性的实数转换成了整数，当数值小于0.5时，相似性记为0，否则记为1；文献[20]将Android应用程序操作码的编码切成两个2位的十六进制数来表示，原本的操作码是以一个位数为4位的十六进制大整数。

3.2.2 模型输入构造

不同的深度学习模型的输入数据结构不同，现有工作所使用的模型对应的输入数据结构主要是序列、向量和矩阵。对于特征数量不固定的系统调用、Android API等，它们都是天然的序列，所以文献[14,18,36,44]将这些特征量化后以序列的数据结构输入到模型中。对于特征数量较固定的特征，通常将它们转换为向量或矩阵，其中向量的长度即为特征数据的个数，如文献[19,21,23,32,36,37]；而矩阵常由一个高维向量拆分而成，如文献[14]，或者由多个向量拼接而成，如文献[11,31]，但是有时某行特征的数目会小于矩阵的列数，这时通常以补0的方式构成矩阵，如文献[31]。

3.3 小结与讨论

尽管深度学习可以从源数据中自动学习特征，但是为了提高模型的检测效果，现有工作基本都提取了Android应用的相关特征作为模型的输入数据。然而，现有工作在提取特征后，通常只是将特征简单数字化，构造组合成模型可接受的数据结构，并未将其与深度学习具体网络结构的特点进行结合，研究导致无法发挥相应网络结构的全部能力。因此，本文建议在未来的研究中，应加强面向深度学习网络结构的特征提取方法研究，以进一步提高深度学习的效果。例如：基于卷积神经网络能够刻画空间关系的特点，可以把输入特征组织成在空间上具有位置关系的形式，而不是简单地将输入特征矩阵化；基于循环神经网络具有记忆性且适配线性输入的特点，可以把输入特征组织成在时间上有先后关系的线性形式并且这种关系应当尽量简单且具有代表性，而不是简单地将输入特征序列化；基于深度信念网络具备适配无向图的特点，可以将输入特征组织成类似于马尔可夫条件随机场[45]这种两两节点之间具有双向关系的无向图的形式，而不是将输入特征简单地变化成一张普通的图。此处提到的神经网络结构将在下一节进行详细介绍。

4 网络结构

随着深度学习的不断发展，神经网络的结构也在不断发展和变化。不同的网络结构有着不同的特性和适用范围。在Android恶意软件检测领域中应用得比较广泛的网络结构大致可以被分为4类：多层感知机(Multi-Layer Perceptron, MLP)、卷积神经网络(Convolutional Neural Network, CNN)、循环神经网络(Recurrent Neural Network, RNN)和深度信念网络(Deep Belief Network, DBN)。下文将分别对这4类的网络结构进行归纳梳理，指出它们在Android恶意软件检测领域之中所遇到的问题，并在最后小结部分提出建议。

4.1 多层感知机

多层感知机神经网络由多层神经元层构成，一般包括输入层、输出层以及在输入与输出层之间的多层隐藏神经元层。同一层的神经元通常互不连接，每一层都全连接到下一层。多层感知机作为一种最基本的网络结构，它在理想情况下可以拟合任意函数[46]，因此学者们将多层感知机引入到Android恶意软件检测领域。下面对本方向的代表文献进行讨论。

多层感知机神经网络包含两个重要参数：隐藏神经元的层数和每层神经元的个数。现有工作[19,21,32,44]就不同的参数选择进行了测试。其中文献[44]测试了8层、16层和32层隐藏神经元层的检测准确率，文献[32]测试了2至4层且每层神经元个数为50, 100, … 500不同组合的准确率，文献[19]测试了1至6层且每层神经元个数为130, 150, 170不同组合的准确率。实验发现并非层数越多、神经元数越多效果便越好。文献[44]发现使用16层和32层时检测效率并

没有显著提高，文献[32]发现含有3层隐藏神经元且每层含有200个神经元的网络结构分类效果最好，文献[19]发现具有2层隐藏神经元层且每层含有150个神经元的网络结构拥有最高的检测准确率。此外，文献[21]则使用一个含有4层隐藏神经元层的网络对Android应用进行分类，该网络的效果最好且其4个隐藏层分别含有250, 200, 150和100个神经元。

在另一个方面，文献[11]尝试构造了一个多模型的网络结构。这个网络结构分为两个部分，第1个部分包含5个相互不连接的并含有2层隐藏神经元层的网络，第2部分则将第1部分的5个网络的输出合并后输入一个同样包含2层隐藏层的网络。这样的多模型网络结构可以将选取模型的任务也交给网络学习过程来自动学习。

从文献[11,19,21,32,44]中，本文归纳发现在将多层感知机神经网络应用到Android恶意软件检测领域的研究中时，大家经过参数搜索后，发现体量相对较小的多层感知机神经网络效果较好，这样的网络一般含有2～4层隐藏层，每层含有100～200个神经元。但是这样的结论是有偏差的，因为多层感知机本身参数量比较大，而参数量大的网络难以收敛；同时，在参数搜索中需要设置一个统一的训练终止条件，这个条件可能是训练迭代次数的上限或者是某个变化率的下限等，这就导致大体量的多层感知机与小体量的网络结构在拥有相同的训练终止条件时，大体量的多层感知机往往是训练不完全的，进而小体量的网络在比较时获胜。此外，以上5份文献并没有结合Android恶意软件所具有的特性来减少网络的参数量，从而较难看到一个经过充分训练后的大体量的网络的效果，进而失去提高准确率的机会。

4.2 卷积神经网络

卷积神经网络于2012年前后兴起于计算机视觉领域[47]。它的特点是利用卷积运算来模拟人观察事物时的聚焦过程，在这个过程中人会聚焦于视野范围内的一个特定区域上，并将这个区域的特征和其他区域进行比较。由于只需要与特定区域的特征进行比较，卷积神经网络大大减少了在网络中需要学习的参数，这不仅提高了网络的学习效率还在很大程度上预防了过拟合的发生。下文将针对网络结构本身，归纳梳理Android恶意代码检测中所使用的多种卷积神经网络结构。

现有工作所使用的卷积神经网络结构主要为LeNet[48]、Inception[49]及它们的演化结构。LeNet结构简单易用，在图片分类问题中它的分类效果往往作为一个基础的标准来同其他工作进行比较。文献[14–16]均利用了LeNet来进行恶意软件检测。文献[39]则使用了一种由LeNet演化而来的网络结构，同时该研究在网络超参的选取上利用了TPE[50]参数搜索方法来寻找一组最适合Android恶意软件检测的超参。Inception结构的特点是将多个不同尺度的卷积核应用于同一个输入上，并将它们的结果合并成输出，从而使网络能同时利用不同尺度的信息来进行决策。文献[31,35]均利用了Inception来进行检测。其中文献[35]使用的是Inception的升级版本Inception-v3[51]。该结构相较于Inception在多个方面进行了优化，不仅提高了训练速度还提高了网络分类的准确率。此外，文献[17]将Lenet和Inception进行了结合，尝试了一种多分支的网络结构。该结构有两个分支网络：CNN-S和CNN-P。其中CNN-S是一种类似于LeNet的结构，包含3个卷积层，当中的最后一个卷积层的卷积核的大小是1×1。而CNN-P是一种类似于Inception的结构，它由3个更小的分支组成。这种多分支的结构也被应用于文献[23]中，且取得了良好的效果。

从上述讨论的工作中可以发现，大多数只是将卷积神经网络直接套用在Android恶意软件检测上。如此仅利用到了卷积神经网络可以有效减少待学习参数量的特性减少训练时间，但是并没有将卷积神经网络所模拟的聚焦过程和Android恶意软件检测这个具体问题相结合，以进一步优化网络结构本身，提高检测准确率。

4.3 循环神经网络

循环神经网络是一种具有序列结构的神经网络，序列上的每一个状态都由当前输入和之前所有状态的累积结果共同决定。这种特殊的结构适用于处理在时间上有先后关系的序列，因此在自然语言处理领域中，循环神经网络被广泛用于建立词语之间的复杂关系，以此来理解整个句子乃至段落的含义。而计算机语言，无论是高级编程语言、汇编语言还是机器码，其本质上也是一门语言，和自然语言的特性有很大的相似性。故而，许多学者将循环神经网络引入到Android恶意软件检测领域中来，意图让其从更高的层次来理解Android恶意软件，从而提高检测准确率。下文将介绍该领域中3个使用循环神经网络的代表工作[14,36,44]。

现有工作[14,36,44]主要使用的循环神经网络是长短期记忆网络(Long Short-Term Memory, LSTM)[52]，该结构具有长期记忆性(可记住长达2000个时间单位之前的状态)。具体而言，文献[14]使用了一个包含有256个单元的双层LSTM将输入数据转化成特征向量，然后将特征向量通过一个全连接分类器进

行分类；文献[44]比较了包含4, 8, 16, 32层LSTM的不同网络结构的检测效果，并最终选择使用了误识率最低的含有32层LSTM的网络；文献[36]则使用了一种特殊的LSTM网络结构即注意力模型[53]，该结构的特点在于可以同时计算出基于当前输入的特征向量和下一个需要注意部分的位置，从而让神经网络更加关注那些具有关键信息的输入，提高模型效果。

从上述研究中可以发现，现有工作已将循环神经网络结构移植到了Android恶意软件检测中。但是，目前这种移植是初步的，利用循环神经网络进行Android恶意软件检测还存在两个严重的问题：(1)Android恶意行为对应的代码可能分布在程序的不同地方，而整个程序时常包含多达上万条的语句，这大大超出了现有的循环神经网络所能记忆的最大尺度；(2)代码指令之间的关系比自然语言词语之间的关系更加复杂，存在着相近语句之间的关系跳跃性更强、关联性更小的特点，通过现有的循环神经网络如LSTM很难完全理解一些指令之间的隐晦关系。

4.4 深度信念网络

深度信念网络由多层受限玻尔兹曼机(restricted Boltzmann machine)组成[54]。较之前所讨论的神经网络所不同的是，深度信念网络的每一层受限玻尔兹曼机是一个无向图，它不是采用通常的梯度下降方法(gradient descent)来训练，而是使用CD(Contrasitive Divergence)方法来训练。CD方法是一种基于采样的方法，旨在找到一组内部参数来最大化输入数据出现的概率。下文将介绍3个使用深度信念网络的代表工作。

文献[19,34,36]便利用了深度信念网络来检测Android恶意代码。其中文献[36]直接应用经典的训练方法来训练深度信念网络进行Android恶意软件检测，而文献[19,34]则使用了一种改进过后的训练方法来训练深度信念网络。具体而言，这个训练方法分为预训练和调整两个步骤。在预训练步骤中，从下到上(从输入层到输出层)的每一层受限玻尔兹曼机都经由非监督的方式来学习数据的分布；在调整步骤中又分为唤醒和生成这两个子步骤，其中唤醒子步骤是从下到上地唤醒每一个神经元从而得到一个数据表示，而生成子步骤则是依据标签与数据表示的差异，从上到下地更新内部参数。

现有工作在使用深度信念网络时或直接套用现有网络结构，或优化了其模型训练算法加速网络训练过程，而忽略了深度信念网络本身所适用的范围。深度信念网络适用于刻画无向图中各节点之间的关系，这有别于Android软件中常常蕴含着的有向图关系，如程序控制关系、数据依赖关系等。现有工作在处理Android软件时缺乏对深度信念网络结构的改进，从而无法全面地理解Android应用，提高检测效率。

4.5 小结与讨论

通过对以上工作的研究，本文发现随着神经网络结构在计算机视觉以及自然语言处理等方面的蓬勃发展，许多网络结构被引入到Android恶意软件检测领域，它们提高了恶意软件检测的准确率和鲁棒性。但是，值得注意的是，面向代码检测领域的网络结构还未被深入研究，学者们大多是简单套用现成的网络结构并稍加改进，而这样的简单套用并不能充分发挥出所选网络结构的优势。因此本文就上述所讨论的几种网络结构如何与Android恶意软件的特性相结合提出几点可以尝试的思路：(1)针对多层感知机神经网络参数多表征潜力大的特点，本文建议可以将多层感知机应用于嵌入内核的次数、代码的匹配程度等抽象层次比较高的输入特征上，这样的特征数量少、信息丰富，可以有效减少多层感知机的规模；(2)针对卷积神经网络能模拟聚焦过程的特性，和Android恶意软件中与恶意行为相关的代码相距位置可能较大的特点，本文建议适当扩大卷积核的感受域(Receptive Field)[55]，并建立浅层特征和深层特征之间长跨度的跳转连接来适配Android软件中的复杂关系；(3)针对循环神经网络的局限性和Android恶意软件中恶意代码相距较远、相邻语句关系复杂的特点，本文建议在应用循环神经网络之前应当先将冗长的代码提炼成具有强语意信息的功能片段，再利用循环神经网络来刻画它们之间的关系；(4)针对深度信念网络适配无向图的特性，本文建议在应用深度信念网络时应当先将输入构建成无向图模型，再依据该无向图模型中的连接关系来精简深度信念网络内部的连接。此外，由于控制流图[56]可以很好地表示软件代码所有可能运行的状态与状态之间的关系，并且基于图的深度学习网络结构的研究正在快速发展[57–60]，因此，本文建议将基于图的深度学习网络结构和Android恶意软件的控制流图相互结合来尝试提高检测效果。

5 检测效果

深度学习是目前在机器学习领域中公认的对分类检测问题效果十分有效的算法。为了评价将深度学习用到Android恶意软件检测领域的效果，现有工作从不同角度对其检测效果的提升进行了衡量。

文献[12,14,19,22,32]使用相同的应用特征，在其实验的数据集上对传统的机器学习模型和深度学

习模型分别进行了测试，实验结果如表3所示。这些工作分别从精准率(precision)、召回率(recall)、F-measure、准确率(accuracy)等指标对它们进行了评测[6)]。从表中可以看到，总的来说相对于传统机器学习模型，深度学习模型在大多数指标上的表现都是最好的；在其中一些工作的测试集上，相较于传统机器学习模型的检测效果，深度学习对某些指标甚至实现了大幅度的提升。

文献[11,20,35,39,44]将它们的检测效果与已有的基于传统机器学习的研究工作的结果进行对比，例如同著名的基于支持向量机的Drebin[28]、基于随机森林的DroidMat[61]等比较。本文对它们的比较结果进行了汇总，见表4所示。值得注意的是，尽管从表中的结果看起来基于深度学习的研究工作效果普遍优于基于传统机器学习的研究工作，但是这些工作使用了不同的应用特征，评测时也采用了不同的测试集，这样的评测对比其实无法用于判断这些工作的优劣。

文献[11,21,32,35,39,44]还将它们自己的检测效果和其他也基于深度学习的Android恶意软件检测研究的检测效果[13,18–20,33,35]进行了对比，如表5所示。经过汇总，本文发现部分研究工作的检测效果在多个文献中差异较大：例如文献[20]其准确率在文献[11]中只有87%，在文献[32]中为98%。不仅如此，在文献[11]的评测中，文献[20]的准确率87%远远低于文献[32]的准确率96.8%，但在文献[32]的评测中，文献[20]的准确率98%却又略高于[19]的准确率96.76%。如此可见，由于缺乏统一的评测数据与模型，导致同一个深度学习模型在不同工作中产生不一样的评测结果。这种不一致性使得学者们无法准确衡量一份工作的优劣。

5.1 小结与讨论

本文通过汇总分析现有工作所报告的检测效果发现，从总体上来说，相较于传统的机器学习模型，基于深度学习的Android恶意软件检测更为有效，其检测效果在各个评价指标上均有提升。同时，本文也发现了现有评测工作中存在的问题，主要有以下两点：(1)缺乏统一的测试集。由于缺乏统一的测试集，表4和表5中的研究工作所给出的比较结果有失客观性，无法准确衡量各个工作之间的优劣。故而，当前亟需一个统一的测试集供该领域的研究工作者在上面进行评测。文献[24,25]工作多年来持续收集、更新了千万数量级的Android应用并对他们进行标定，本文建议未来的的研究工作可以在这些开放数据集上进行评测。(2)缺乏标准的评价指标。本文从上述分析中还发现，评价机器学

表 3　在相同数据下现有深度学习模型与传统机器学习模型效果对比统计表(%)

研究工作	评价指标	深度学习模型	传统机器学习模型					
			支持向量机	决策树	朴素贝叶斯	逻辑回归	随机森林	K最近邻
文献[12]	m4	**96.5**	80.0	77.5	79.0	78.0		
文献[14]	m1	**100**	53.3		47.0			
	m2	**98.3**	34.8		54.0			
	m4	**99.4**	66.0		82.0			
文献[19]	m1	**95.77**	92.08	75.09	79.22	64.18		
	m2	97.84	93.75	**98.64**	91.82	95.91		
	m4	**96.76**	92.84	82.95	83.86	71.19		
文献[22]	m1	**99.52**	94.23	93.77		95.64	97.04	95.40
	m2	**99.83**	95.89	94.68		95.90	94.69	93.16
	m3	**99.74**	95.05	94.22		95.77	95.85	94.27
	m4	**99.68**	94.97	94.13		95.82	95.93	94.29
文献[32]	m1	**94.82**	87.6	92	76.5		93.8	
	m2	**97.76**	87.5	92	76.8		93.8	
	m5	90.86	94.4	**95.5**	85.5		97.1	
	m6	9.14	5.6	4.5	14.5		**2.9**	
	m7	**2.24**	24.2	13.9	38		12	

注：各评价指标的含义如下。m1：精确率(Precision)，m2：召回率/真正率(recall/TPR)，m3：F-measure，m4：准确率(accuracy)，m6：假正率(FPR)，m7：假负率(FNR)

[6)]在表3、表4和表5中，加粗条目表示该指标下最优异的测试结果。

表 4 在不同数据不同特征下现有基于深度学习的方法与基于传统机器学习的方法效果对比统计表

研究工作	机器学习模型	m1(%)	m2(%)	m3(%)	m4(%)	m6(%)	m7(%)	m8(s)
文献[11]	深度学习			**98**	**99**			
文献[28]	支持向量机			93.9				
文献[62]	决策树				78			
文献[63]	朴素贝叶斯			93				
文献[61]	K最近邻				99			
文献[64]	极限梯度提升决策树			97	97			
文献[35]	深度学习		**96**		**93**	9		**0.5**
文献[28]	支持向量机		94.0			1.0		0.75
文献[65]	随机森林		95.3		92	**0.34**		19.8
文献[39]	深度学习	**98.84**	**98.47**	**98.65**	**98.86**			
文献[66]	逻辑回归	80.99	87.11	83.93	83.26			
文献[44]	深度学习			**98.98**		**1.58**		
文献[67]	随机森林			97.42		4.33		
文献[20]	深度学习	**99**	**95**	97	**98**			
文献[68]	支持向量机			**98**				
文献[69]	朴素贝叶斯	94	91	92	91			
文献[67]	随机森林	98	97	97	97			

注：各评价指标的含义如下。m1：精确率(Precision)，m2：召回率/真正率(recall/TPR)，m3：F-measure，m4：准确率(accuracy)，m6：假正率(FPR)，m7：假负率(FNR)，m8：检测时间

表 5 基于深度学习的Android恶意软件检测工作效果互相对比统计表(%)

研究工作	m1	m2	m3	m4	m6	m7
文献[11]			**99**	**98**		
文献[19]				96.8		
文献[20]			86	87		
文献[35]		96		93	9	**0.5**
文献[20]		**99.3**		**99**	**3**	2.5
文献[39]	**98.87**	**98.47**	**98.65**	**98.86**		
文献[13]	83.24	87.67	85.39	84.95		
文献[18]	94.76	91.31	93.00	93.10		
文献[20]	67	**98.47**	71.00	69.00		
文献[44]			**98.98**		**1.58**	
文献[20]			89.50		6.72	
文献[32]	98.09	**99.56**	**98.82**	**98.5**		
文献[33]	93.96	93.36	93.68	93.68		
文献[19]	96.78	96.76	96.76	96.76		
文献[20]	**99**	95	97	98		
文献[21]				**95.31**		
文献[35]				93		
文献[33]				93.68		

注：各评价指标的含义如下。m1：精确率(Precision)，m2：召回率/真正率(recall/TPR)，m3：F-measure，m4：准确率(accuracy)，m6：假正率(FPR)，m7：假负率(FNR)

习模型效果的指标很繁杂，不同工作所使用的评价指标也各不相同。鉴于Android恶意软件检测可归纳为二分问题，即把一个Android应用分为良性软件或恶意软件，因此本文建议未来该领域的相关

工作可采用机器学习二分工作通常采用的精确率(precision)、召回率(recall)以及F-measure这3个主要指标[70]进行模型评价。

6　结束语

本文从数据采集、应用特征、网络结构和效果检测这4个方面，对基于深度学习的Android恶意软件检测现有的学术成果进行分析与研究。可以发现该方向的研究已取得了若干显著进展：在采集的海量Android应用数据集的支撑下，挖掘出了种类更丰富的特征去更加准确地、全面地描述Android恶意软件的特点；同时，相较于基于传统机器学习的检测方法，基于深度学习的检测方法提高了Android恶意软件检测效果，在各个评价指标上均有提升。当然，现有工作也存在一些不足，其中最主要的表现在两个方面：所提取的Android应用特征及其组织方式未结合具体深度学习网络结构的特点，无法发挥这些网络结构的全部能力；所采用的网络结构也未面向Android恶意软件检测的特性进行深入设计，其网络结构在形式上和理论上均不具备完全理解Android恶意软件的能力。因此，研究提取构造更适应于网络结构的应用特征和研究设计能更好理解Android应用特性的网络结构是未来该领域的两个重要研究方向。期待更多学者加入到基于深度学习的Android恶意软件检测研究中，使得该方向的研究成果能在实际检测中发挥作用。

参 考 文 献

[1] CHAU M and REITH R. Smartphone market share[EB/OL]. https://www.idc.com/promo/smartphone-market-share/os, 2019.

[2] Tencent Mobile Butler. Tencent mobile security lab mobile security report in the first half year of 2019[EB/OL]. https://m.qq.com/security_lab/news_detail_517.html, 2019.

[3] WANG Bolun, YAO Yuanshun, SHAN S, *et al*. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks[C]. 2019 IEEE Symposium on Security and Privacy, San Francisco, USA, 2019: 707–723. doi: 10.1109/SP.2019.00031.

[4] SAFAVIAN S R and LANDGREBE D. A survey of decision tree classifier methodology[J]. *IEEE Transactions on Systems, Man, and Cybernetics*, 1991, 21(3): 660–674. doi: 10.1109/21.97458.

[5] SUYKENS J A K and VANDEWALLE J. Least squares support vector machine classifiers[J]. *Neural Processing Letters*, 1999, 9(3): 293–300. doi: 10.1023/A:1018628609742.

[6] MCCALLUM A and NIGAM K. A comparison of event models for naive Bayes text classification[C]. AAAI-98 Workshop on Learning for Text Categorization, Madison, Isconsin, USA, 1998: 41–48.

[7] 王鑫, 李可, 宁晨, 等. 基于深度卷积神经网络和多核学习的遥感图像分类方法[J]. 电子与信息学报, 2019, 41(5): 1098–1105. doi: 10.11999/JEIT180628.
WANG Xin, LI Ke, NING Chen, *et al*. Remote sensing image classification method based on deep convolution neural network and multi-kernel learning[J]. *Journal of Electronics & Information Technology*, 2019, 41(5): 1098–1105. doi: 10.11999/JEIT180628.

[8] 徐少平, 张贵珍, 李崇禧, 等. 基于深度置信网络的随机脉冲噪声快速检测算法[J]. 电子与信息学报, 2019, 41(5): 1130–1136. doi: 10.11999/JEIT180558.
XU Shaoping, ZHANG Guizhen, LI Chongxi, *et al*. A fast random-valued impulse noise detection algorithm based on deep belief network[J]. *Journal of Electronics & Information Technology*, 2019, 41(5): 1130–1136. doi: 10.11999/JEIT180558.

[9] 杨宏宇, 王峰岩. 基于深度卷积神经网络的气象雷达噪声图像语义分割方法[J]. 电子与信息学报, 2019, 41(10): 2373–2381. doi: 10.11999/JEIT190098.
YANG Hongyun and WANG Fengyan. Meteorological radar noise image semantic segmentation method based on deep convolutional neural network[J]. *Journal of Electronics & Information Technology*, 2019, 41(10): 2373–2381. doi: 10.11999/JEIT190098.

[10] STATISTA. Number of available applications in the Google Play Store from December 2009 to June 2020[EB/OL]. https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/, 2019.

[11] KIM T, KANG B, RHO M, *et al*. A multimodal deep learning method for Android malware detection using various features[J]. *IEEE Transactions on Information Forensics and Security*, 2019, 14(3): 773–788. doi: 10.1109/TIFS.2018.2866319.

[12] YUAN Zhenlong, LU Yongqiang, WANG Zhaoguo, *et al*. Droid-sec: Deep learning in android malware detection[C]. 2014 ACM Conference on SIGCOMM, Chicago, USA, 2014: 371–372. doi: 10.1145/2619239.2631434.

[13] XIAO Xi, WANG Zhenlong, LI Qing, *et al*. Back-propagation neural network on Markov chains from system call sequences: A new approach for detecting Android malware with system call sequences[J]. *IET Information Security*, 2017, 11(1): 8–15. doi: 10.1049/iet-ifs.2015.0211.

[14] NIX R and ZHANG Jian. Classification of Android apps and malware using deep neural networks[C]. 2017 International Joint Conference on Neural Networks, Anchorage, USA, 2017: 1871–1878. doi: 10.1109/IJCNN.

2017.7966078.

[15] HUANG Na, XU Ming, ZHENG Ning, *et al*. Deep android malware classification with API-based feature graph[C]. The 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering, Rotorua, New Zealand, 2019: 296–303. doi: 10.1109/TrustCom/BigDataSE.2019.00047.

[16] ABDERRAHMANE A, ADNANE G, YACINE C, *et al*. Android malware detection based on system calls analysis and CNN classification[C]. 2019 IEEE Wireless Communications and Networking Conference Workshop, Marrakech, Morocco, 2019: 1–6. doi: 10.1109/WCNCW.2019.8902627.

[17] WANG Wei, ZHAO Mengxue, and WANG Jigang. Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network[J]. *Journal of Ambient Intelligence and Humanized Computing*, 2019, 10(8): 3035–3043. doi: 10.1007/s12652-018-0803-6.

[18] XIAO Xi, ZHANG Shaofeng, MERCALDO F, *et al*. Android malware detection based on system call sequences and LSTM[J]. *Multimedia Tools and Applications*, 2019, 78(4): 3979–3999. doi: 10.1007/s11042-017-5104-0.

[19] YUAN Zhenlong, LU Yongqiang, and XUE Yibo. Droiddetector: Android malware characterization and detection using deep learning[J]. *Tsinghua Science and Technology*, 2016, 21(1): 114–123. doi: 10.1109/TST.2016.7399288.

[20] MCLAUGHLIN N, MARTINEZ DEL RINCON J, KANG B, *et al*. Deep android malware detection[C]. The 7th ACM on Conference on Data and Application Security and Privacy, Scottsdale, USA, 2017: 301–308. doi: 10.1145/3029806.3029823.

[21] NAWAY A and LI Yuancheng. Using deep neural network for Android malware detection[EB/OL]. https://arxiv.org/pdf/1904.00736, 2019.

[22] WANG Zhiqiang, LI Gefei, CHI Yaping, *et al*. Android malware detection based on convolutional neural networks[C]. The 3rd International Conference on Computer Science and Application Engineering, Sanya, China, 2019: 1–151. doi: 10.1145/3331453.3361306.

[23] SABHADIYA S, BARAD J, and GHEEWALA J. Android malware detection using deep learning[C]. The 3rd International Conference on Trends in Electronics and Informatics, Tirunelveli, India, 2019: 1254–1260. doi: 10.1109/ICOEI.2019.8862633.

[24] MELISSA. VirusShare. Com-because sharing is caring[EB/OL]. https://virusshare.com, 2019.

[25] ALLIX K, BISSYANDÉ T F, KLEIN J, *et al*. Androzoo: Collecting millions of android apps for the research community[C]. The 13th IEEE/ACM Working Conference on Mining Software Repositories, Austin, TX, USA, 2016: 468–471.

[26] WEI Fengguo, LI Yuping, ROY S, *et al*. Deep ground truth analysis of current android malware[C]. The 14th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Bonn, Germany, 2017: 252–276. doi: 10.1007/978-3-319-60876-1.

[27] MILAPARKOUR. Contagio mobile mobile malware mini dump[EB/OL]. http://contagiominidump.blogspot.com, 2019.

[28] ARP D, SPREITZENBARTH M, HUBNER M, *et al*. Drebin: Effective and explainable detection of android malware in your pocket[C]. The 21st Annual Network and Distributed System Security Symposium, San Diego, California, USA, 2014: 23–26. doi: 10.14722/ndss.2014.23247.

[29] KADIR A F A, STAKHANOVA N, and GHORBANI A A. Android botnets: What URLs are telling us[C]. The 9th International Conference on Network and System Security, New York, NY, 2015: 78–79. doi: 10.1007/978-3-319-25645-0_6.

[30] ZHOU Yajin and JIANG Xuxian. Dissecting android malware: Characterization and evolution[C]. 2012 IEEE Symposium on Security and Privacy, San Francisco, USA, 2012: 95–109. doi: 10.1109/SP.2012.16.

[31] YE Yanfang, HOU Shifu, CHEN Lingwei, *et al*. Out-of-sample node representation learning for heterogeneous graph in real-time android malware detection[C]. The 28th International Joint Conference on Artificial Intelligence, Macao, China, 2019: 4150–4156. doi: 10.24963/ijcai.2019/576.

[32] ALZAYLAEE M K, YERIMA S Y, and SEZER S. DL-Droid: Deep learning based android malware detection using real devices[J]. *Computers & Security*, 2020, 89: 101663. doi: 10.1016/j.cose.2019.101663.

[33] HOU Shifu, SAAS A, CHEN Lifei, *et al*. Deep4MalDroid: A deep learning framework for android malware detection based on Linux kernel system call graphs[C]. 2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops, Omaha, USA, 2016: 104–111. doi: 10.1109/WIW.2016.040.

[34] HOU Shifu, SAAS A, YE Yanfang, *et al*. Droiddelver: An android malware detection system using deep belief network based on API call blocks[C]. WAIM 2016 International Conference on Web-Age Information Management, Nanchang, China, 2016: 54–55. doi: 10.1007/978-3-319-

47121-1_5.

[35] HUANG T H D and KAO H Y. R2-D2: Color-inspired convolutional neural network (CNN)-based android malware detections[C]. 2018 IEEE International Conference on Big Data, Seattle, USA, 2018: 2633–2642. doi: 10.1109/BigData.2018.8622324.

[36] NAUMAN M, TANVEER T A, KHAN S, *et al*. Deep neural architectures for large scale android malware analysis[J]. *Cluster Computing*, 2018, 21(1): 569–588. doi: 10.1007/s10586-017-0944-y.

[37] DUC N V and GIANG P T. NADM: Neural network for Android detection malware[C]. The 9th International Symposium on Information and Communication Technology, Danang City, Vietnam, 2018: 449–455. doi: 10.1145/3287921.3287977.

[38] AAFER Y, DU WENLIANG, and YIN Heng. Droidapiminer: Mining API-level features for robust malware detection in android[C]. The 9th International Conference on Security and Privacy in Communication Systems, Sydney, Australia, 2013: 86–103. doi: 10.1007/978-3-319-04283-1_6.

[39] PEKTAŞ A and ACARMAN T. Deep learning for effective Android malware detection using API call graph embeddings[J]. *Soft Computing*, 2020, 24(2): 1027–1043. doi: 10.1007/s00500-019-03940-5.

[40] SUN Yizhou and HAN Jiawei. Mining heterogeneous information networks: Principles and methodologies[J]. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 2012, 3(2): 1–159. doi: 10.2200/S00433ED1V01Y201207DMK005.

[41] FAN Yujie, HOU Shifu, ZHANG Yiming, *et al*. Gotcha-sly malware!: Scorpion a metagraph2vec based malware detection system[C]. The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 2018: 253–262. doi: 10.1145/3219819.3219862.

[42] DONG Yuxiao, CHAWLA N V, and SWAMI A. Metapath2vec: Scalable representation learning for heterogeneous networks[C]. The 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, Canada, 2017: 135–144. doi: 10.1145/3097983.3098036.

[43] FU Taoyang, LEE W C, and LEI Zhen. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning[C]. 2017 ACM on Conference on Information and Knowledge Management, Singapore, 2017: 1797–1806. doi: 10.1145/3132847.3132953.

[44] MA Zhuo, GE Haoran, LIU Yang, *et al*. A combination method for Android malware detection based on control flow graphs and machine learning algorithms[J]. *IEEE Access*, 2019, 7: 21235–21245. doi: 10.1109/ACCESS.2019.2896003.

[45] GEORGE R C and JAIN A K. Markov random field texture models[J]. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1983, PAMI-5(1): 25–39. doi: 10.1109/TPAMI.1983.4767341.

[46] CSÁJI B C. Approximation with artificial neural networks[D]. [Master dissertation], Eotvos Loránd University, 2001: 7.

[47] KRIZHEVSKY A, SUTSKEVER I, and HINTON G E. ImageNet classification with deep convolutional neural networks[C]. The 25th International Conference on Neural Information Processing Systems, Lake Tahoe, USA, 2012: 1106–1114.

[48] LECUN Y, BOTTOU L, BENGIO Y, *et al*. Gradient-based learning applied to document recognition[J]. *Proceedings of the IEEE*, 1998, 86(11): 2278–2324. doi: 10.1109/5.726791.

[49] SZEGEDY C, LIU Wei, JIA Yangqing, *et al*. Going deeper with convolutions[C]. The 2015 IEEE Conference on Computer Vision and Pattern Recognition, Boston, USA, 2015: 1–9. doi: 10.1109/CVPR.2015.7298594.

[50] BERGSTRA J, BARDENET R, BENGIO Y, *et al*. Algorithms for hyper-parameter optimization[C]. The 25th Annual Conference on Neural Information Processing Systems 2011, Granada, Spain, 2011: 2546–2554.

[51] SZEGEDY C, VANHOUCKE V, IOFFE S, *et al*. Rethinking the inception architecture for computer vision[C]. 2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, USA, 2016: 2818–2826. doi: 10.1109/CVPR.2016.308.

[52] HOCHREITER S and SCHMIDHUBER J. Long short-term memory[J]. *Neural Computation*, 1997, 9(8): 1735–1780. doi: 10.1162/neco.1997.9.8.1735.

[53] MNIH V, HEESS N, GRAVES A, *et al*. Recurrent models of visual attention[C]. The 27th International Conference on Neural Information Processing Systems, Montreal, Canada, 2014: 2204–2212.

[54] SALAKHUTDINOV R and MURRAY I. On the quantitative analysis of deep belief networks[C]. The 25th International Conference on Machine Learning, Helsinki, Finland, 2008: 872–879. doi: 10.1145/1390156.1390266.

[55] ALONSO J M and CHEN Yao. Receptive field[J]. *Scholarpedia*, 2009, 4(1): 5393. doi: 10.4249/scholarpedia.5393.

[56] ALLEN F E. Control flow analysis[J]. *ACM SIGPLAN Notices*, 1970, 5(7): 1–19. doi: 10.1145/390013.808479.

[57] SCARSELLI F, GORI M, TSOI A C, *et al*. The graph neural network model[J]. *IEEE Transactions on Neural Networks*, 2009, 20(1): 61–80. doi: 10.1109/TNN.2008.

2005605.

[58] JIANG Bo, ZHANG Ziyan, LIN Doudou, *et al*. Semi-supervised learning with graph learning-convolutional networks[C]. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, USA, 2019: 11305–11312. doi: 10.1109/CVPR.2019.01157.

[59] MARON H, BEN-HAMU H, SERVIANSKY H, *et al*. Provably powerful graph networks[C]. The 33rd Conference on Neural Information Processing Systems, Vancouver, Canada, 2019: 2153–2164.

[60] GORI M, MONFARDINI G, and SCARSELLI F. A new model for learning in graph domains[C]. 2005 IEEE International Joint Conference on Neural Networks, Montreal, Canada, 2005: 729–734.

[61] WU Dongjie, MAO C H, WEI T E, *et al*. Droidmat: Android malware detection through manifest and API calls tracing[C]. The 7th Asia Joint Conference on Information Security, Tokyo, Japan, 2012: 62–69. doi: 10.1109/AsiaJCIS.2012.18.

[62] HUANG C Y, TSAI Y T, and HSU C H. Performance evaluation on permission-based detection for android malware[C]. International Computer Symposium ICS 2012 Held at Hualien, Taipei, China, 2013: 111–120. doi: 10.1007/978-3-642-35473-1_12.

[63] ZHANG Mu, DUAN Yue, YIN Heng, *et al*. Semantics-aware android malware classification using weighted contextual API dependency graphs[C]. 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, USA, 2014: 1105–1116. doi: 10.1145/2660267.2660359.

[64] FEREIDOONI H, CONTI M, YAO Danfeng, *et al*. ANASTASIA: Android malware detection using static analysis of applications[C]. The 8th IFIP International Conference on New Technologies, Mobility and Security, Larnaca, Cyprus, 2016: 1–5. doi: 10.1109/NTMS.2016.7792435.

[65] YANG Chao, XU Zhaoyan, GU Guofei, *et al*. Droidminer: Automated mining and characterization of fine-grained malicious behaviors in android applications[C]. The 19th European Symposium on Research in Computer Security, Wroclaw, Poland, 2014: 163–182. doi: 10.1007/978-3-319-11203-9_10.

[66] DIMJAŠEVIĆ M, ATZENI S, UGRINA I, *et al*. Evaluation of android malware detection based on system calls[C]. 2016 ACM on International Workshop on Security and Privacy Analytics, New Orleans, USA, 2016: 1–8. doi: 10.1145/2875475.2875487.

[67] YERIMA S Y, SEZER S, and MUTTIK I. High accuracy android malware detection using ensemble learning[J]. *IET Information Security*, 2015, 9(6): 313–320. doi: 10.1049/iet-ifs.2014.0099.

[68] JEROME Q, ALLIX K, STATE R, *et al*. Using opcode-sequences to detect malicious Android applications[C]. 2014 IEEE International Conference on Communications, Sydney, Australia, 2014: 914–919. doi: 10.1109/ICC.2014.6883436.

[69] YERIMA S Y, SEZER S, MCWILLIAMS G, *et al*. A new android malware detection approach using Bayesian classification[C]. The 27th IEEE International Conference on Advanced Information Networking and Applications, Barcelona, Spain, 2013: 121–128. doi: 10.1109/AINA.2013.88.

[70] POWERS D M W. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation[J]. *Journal of Machine Learning Technologies*, 2011, 2(1): 37–63.

陈 怡：1991年生，博士生，研究方向为移动应用安全、漏洞挖掘.
唐 迪：1991年生，博士生，研究方向为基于机器学习的安全研究.
邹 维：1964年生，研究员，博士生导师，研究方向为网络与软件安全.

责任编辑：陈 倩