# 基于 N-gram 的 Android 恶意检测

章宗美[1] 桂盛霖[1,2] 任 飞[2]
(电子科技大学计算机科学与工程学院 成都 611731)[1]
(中国电子科技集团公司第三十研究所 成都 610041)[2]

**摘 要** 随着 Android 系统的广泛应用,Android 平台下的恶意应用层出不穷,并且恶意应用躲避现有检测工具的手段也越来越复杂,亟需更有效的检测技术来分析恶意行为。文中提出并设计了一种基于 N-gram 的静态恶意检测模型,该模型通过逆向手段反编译 Android APK 文件,利用 N-gram 技术在字节码上提取特征,以此避免传统检测中专家知识的依赖。同时,该模型使用深度置信网络,能够快速而准确地学习训练。通过对 1267 个恶意样本和 1200 个善意样本进行测试,结果显示模型整体的检测准确率最高可以达到 98.34%。实验进一步比较了该模型和其他算法的检测结果,并对比了相关工作的检测效果,结果表明该模型有更好的准确率和鲁棒性。

**关键词** Android 应用,恶意检测,N-gram,深度置信网络,静态检测
**中图法分类号** TP309.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.02.023

## Android Malware Detection Based on N-gram

ZHANG Zong-mei[1] GUI Sheng-lin[1,2] REN Fei[2]
(School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China)[1]
(The 30th Institute of China Electronics Technology Group Corporation, Chengdu 610041, China)[2]

**Abstract** With the widespread use of Android operating system, malicious applications are constantly emerging on the Android platform, meanwhile, the means by which malicious applications evade existing detection tools are becoming increasingly complicated. In order to effectively analyze malicious behavior, more efficient detection technology is required. This paper presented and designed a static malicious detection model based on N-gram technology. The model decompiles Android APK files by reversing engineering and uses N-gram technology to extract features from bytecodes. In this way, the model avoids dependence on expert knowledge in traditional detection. At the same time, the model combines with deep belief network, which allows it to rapidly and accurately train and detect application samples. 1267 malicious samples and 1200 benign samples were tested. The results show that the overall accuracy is up to 98.34%. Furthermore, the results of the model were compared with those of other machine learning algorithms, and the detection results of the related work were also compared. The results show that the model has better accuracy and robustness.

**Keywords** Android application, Malware detection, N-gram, Deep belief network, Static detection

## 1 引言

近年来,随着我国 3G 网络、4G 网络等移动互联网基础设施的完善,智能手机普及率逐年上升,智能手机目前已成为人们日常生活中密不可分的一个部分。Android 系统则凭借其开源、高效的特性被手机厂商以及开发者关注,现已是市场上占有率最高的移动操作系统[1]。但是 Android 平台安全事件层出不穷,恶意应用肆意泛滥,各种安全问题[2]时有发生,不但对用户的生活带来了诸多安全隐患,更影响了移动互联网行业的可信度,阻碍了行业的健康发展。因此,Android 平台下恶意应用的检测与分析成为了国内外重要的研究课题。

Android 平台下的恶意检测主要分为动态检测[3]和静态检测[3]两类技术。动态检测是指将应用运行在沙箱或者虚拟机中,记录下应用在运行时的行为特征,对动态特征进行分析,判断应用是否含有恶意行为;静态检测则在不运行应用程序的情况下,通过逆向工程手段反编译 Android APK 文件,分析并抽取出 Android 应用的特征,如应用申请的权限、应用调用的系统框架 API 等,对抽取出的特征进行分析,从而判断应用是否含有恶意行为。动态检测记录了应用的运行时行为,因此检测精度高,但是动态检测所需的测试环境的资源消耗大,检测效率低,检测环境实现复杂。相对于动态检测来说,静态检测不需要运行应用,资源消耗低,适用于大规模检测。

到稿日期:2018-01-18 返修日期:2018-03-24 本文受国家自然科学基金(61401067)资助。
**章宗美**(1993-),男,硕士生,主要研究方向为 Android 安全研究,E-mail:zach_41@163.com(通信作者);**桂盛霖**(1983-),男,博士,副教授,CCF 会员,主要研究方向为网络空间安全,E-mail:shenglin_gui@uestc.edu.cn(通信作者);**任 飞**(1983-),男,硕士,高级工程师,主要研究方向为密码算法。

在当前 Android 平台的恶意检测研究中，越来越多的研究人员将机器学习和静态检测相结合，并提出了许多基于应用行为特征的方法，这些方法大多使用 Android 的应用权限、API 调用接口等来辅助分析，在很大程度上依赖于 Android 安全领域知识来决定哪些权限或者是 API 作为应用的特征。本文提出了一种基于 N-gram 技术的 Android 恶意应用检测技术，该技术自动化地选取编译之后的字节码的特征而不需要 Android 的专家知识，同时结合深度学习的方法来检测恶意应用程序。实验结果表明其检测的准确率最高可以达到 98.34%。

## 2 相关工作

Android 静态恶意检测主要有基于签名的检测、基于应用特征的检测以及基于控制流图的检测 3 类技术。Androguard[4] 是国外著名的恶意检测工具，它基于开发者签名对应用进行恶意检测。Androguard 通过查询应用的签名是否存在于恶意签名库中来判断应用是否存在恶意行为。这种方法的优点在于检测速度快，准确率高，但是无法检测出未知的恶意应用。

在基于应用特征的检测技术方面，文献[5]提出了一种基于 Dalvik 指令的 Android 恶意代码特征的形式化描述和分析方法，不需要反编译应用程序便可以快速检测样本的恶意特征，但是该方法需要通过人工分析来提取恶意代码片段，检测效率较低。文献[6]设计了一种针对 Android 权限的检测机制，该机制根据权限使用的频率设置权限组，对不同的权限组赋予不同的权值，并利用 K-means 算法进行聚类分析。文献[7]则提出了权限和 API 特征相结合的 Android 恶意软件检测方法，提取了 API 和权限特征，以这些特征相结合的方式对恶意软件进行分类检测。文献[8]提出的 Androdect 工具提取了 Android 组件、函数调用以及系统调用的 3 类特征，同时设计了三层混合系算法 THEA，以检验应用是否具有恶意行为。文献[9]将应用程序调用的 Android 敏感 API 作为特征集合，并结合深度置信网络(Deep Belief Network)[10] 构建恶意检测模型。本文较文献[5-9]的主要区别在于本文的检测模型直接在 Android 字节码上提取应用特征，不需要 Android 安全领域知识，减少了人工分析的工作，并且本文利用 DBN 算法充分挖掘了应用程序更深层次的特征，可以更有效地进行恶意检测。

基于控制流图的恶意检测技术对 Android 应用进行逆向分析，构建应用程序执行过程的控制流图，以此分析应用程序中是否存在隐私数据的泄露。这类技术的难点在于如何准确地构建 Android 应用组件的各个生命周期。文献[11]提出了 Android 平台下的污点分析工具 LeakMiner，LeakMiner 分析了 Android 各个组件的生命周期，并构建了一个能够覆盖 Android 所有回调方法的模型，但是没有考虑污点数据的上下文敏感性，无法检测出隐式的隐私数据泄露。文献[12]则在文献[11]的基础上提出了一个上下文敏感的污点检测模型，但是该模型不具备域敏感以及对象敏感特性，如果一个对象的某一个属性存储了污点数据，那么模型就会判定整个对象为污点数据，导致污点范围扩大，从而导致检测误差。文献[13]提出的 FlowDroid 污点检测工具构建了覆盖 Android 所有组件生命周期及控件回调函数的控制流图，并且具备上下文、域以及对象敏感，同时 FlowDroid 使用 IFDS 框架[14] 对污点数据进行分析。该模型实现了 Android 组件内污点数据传播的分析检测，但是无法分析组件间的污点分析检测。针对文献[13]的问题，文献[15]提出了污点分析工具 Epicc，它在 FlowDroid 的基础上进行了修改和优化，能够有效地检测出 Android 组件间的隐私数据传播。

本文提出的方法是基于 N-gram 技术的，该技术的优点在于直接从 Dalvik 字节码上提取应用特征，而不需要 Android 安全领域的专家知识。文献[16]也提出了一种基于 N-gram 技术的 PC 平台下的恶意检测技术，该技术直接在应用文件上使用 N-gram 提取特征，并结合机器学习方法完成恶意检测。其中，PC 端应用程序由 X86 汇编代码组成，而 Android 应用反编译之后得到的是包含 Dalvik 字节码指令的代码文件，两者的差别较大，因此在特征提取时必须采用不同的方式来进行处理。另一方面，文献[16]仅采用了简单的机器学习模型来对提取的特征进行检测分析，并没有利用 DBN 等神经网络模型来挖掘更深层次的特征信息。本文结合深度置信网络来快速而准确地提取 Android 应用深层次的特征信息，从而建立一个稳定的检测模型。

## 3 检测模型设计

本文提出了一个直接在 Android 的 Dalvik 字节码上进行检测分析的恶意应用检测模型，该模型基于 N-gram 方法，在字节码上直接提取应用的特征，并结合深度置信网络(DBN)对样本进行训练分析。模型的整体设计如图 1 所示。模型主要分为 4 个部分：恶意特征集合的选取、训练集合的向量化、测试集合的向量化、深度置信网络训练模型。

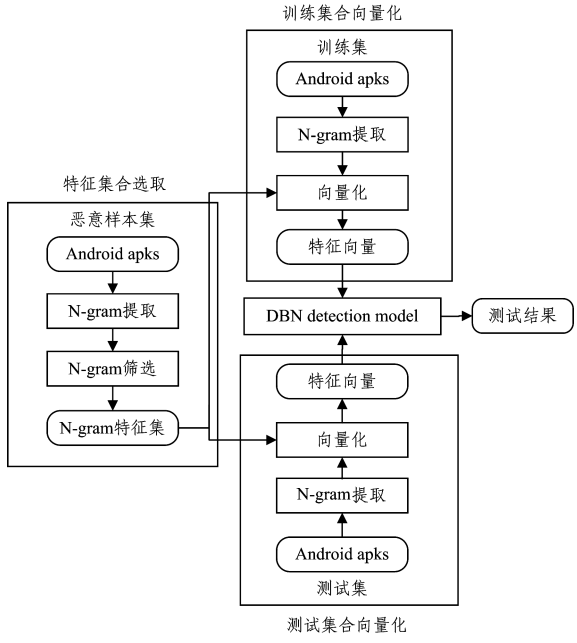


图 1 检测模型的设计

Fig. 1 Design of detection model

恶意特征集合的提取利用下文介绍的 N-gram 提取流程解析所有的恶意样本，得到所有恶意样本的 N-gram 集合 $S$，

分析 $S$ 并选取最能标识恶意应用的 N-gram 子集 $MSet$ 作为恶意特征集合。

$$MSet \subset S \tag{1}$$

其中，$|MSet|$ 为特征向量的维数。训练样本集和测试样本集的向量化首先将样本利用 N-gram 提取流程得到样本的 N-gram 集合，再按照第 3.2 节所述的方法对恶意应用特征集合进行向量化，以此得到一个应用的向量表示。本文选取善意样本集合的 90%和恶意样本集合的 90%作为训练集合，其余作为测试集合。在得到所有测试样本和训练样本的特征向量后，利用 DBN 算法模型进行训练和测试，从而得到检测结果。

### 3.1　N-gram 提取

N-gram 的提取流程如图 2 所示，提取的步骤如下：

1）利用逆向工具 Apktool[17] 反编译 Android 应用；

2）扫描反编译得到的所有. smali 文件，按照转换规则对每一条 Dalvik 字节码指令进行转换；

3）根据 N-gram 分割规则处理步骤 2）得到的转换序列，从而获得样本的 N-gram 集合。

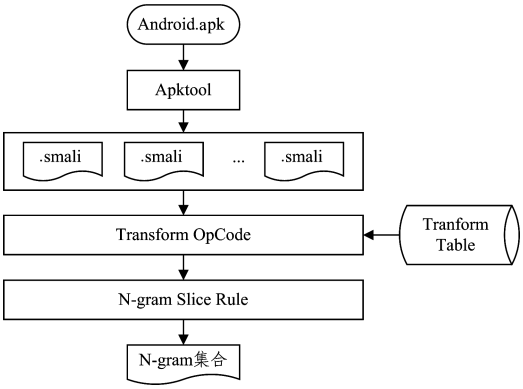


图 2　N-gram 提取流程

Fig. 2　Process of extracting N-gram

一个 Android 的. apk 文件实际上是一个压缩包文件，包含了 Manifest 资源文件、应用资源和可执行的 Dalvik(dex)等文件。利用 apktool 可以解包. apk 文件并反编译. dex 文件，得到包含应用字节码的所有. smali 文件，每一个. smali 文件是一个类的源码文件。每一条 Dalvik 指令包含指令操作、指令操作数以及类型信息，在转换过程中，舍弃指令的操作数以及类型信息，然后在转换表中查找指令对应的序号，依次得到指令所对应的序列。表 1 列出了 Dalvik 指令转换表的部分内容。

表 1　Dalvik 指令转换表(部分)

Table 1　Part of Dalvik instruction transforming table

| Dalvik 指令 | 转换后的序号 |
|---|---|
| nop | 00 |
| move | 01 |
| move/from16 | 02 |
| move/16 | 03 |
| move-wide | 04 |
| move-wide/from16 | 05 |
| invoke-virtual | 6e |
| invoke-super | 6f |
| move-object | 07 |
| new-instance | 22 |
| iput-object | 5b |
| return-void | 0e |

```
invoke-virtual {p0}, L/MainClass;->test()V
invoker-super {p0, p1}, L/Super;->superTest()V
move-object v0, p0
new-instance v1, L/TestClass$1;
iput-object v1, p0, L/MainClass;->test2()V;
return-void
```

图 3　Dalvik 指令段样例

Fig. 3　Example of Dalvik instruction block

以图 3 给出的 Dalvik 字节码指令段为例，舍弃指令的操作数和类型信息，得到{*invoke-virtual*，*invoker-super*，*move-object*，*new-instance*，*iput-object*，*return-void*}，根据转换表依次将指令转换成序列值，得到序列集合{6e,6f,07,22,5b,0e}。

假设 N-gram 的长度为 $N$，那么切割的方式为从序列开始处，以步长为 1，依次取 $N$ 个 gram 得到相应的 N-gram 集合。例如，对前文所得到的指令序列号集合采用长度为 3-gram 的方式进行切割，得到的 N-gram 集合为{6e6f07，6f0722，07225b，225b0e}。

对于指令长度为 $N$ 的 N-gram，假设 Dalvik 指令集的指令个数为 $M$，那么不同的 N-gram 数量的理论上限为 $M^N$。随着 N-gram 长度 $N$ 值的增大，N-gram 提取所得的不同的 N-gram 数量可能会呈现爆炸式增长，从而导致样本的特征向量维数太大，训练和匹配阶段的计算时间过长，检测效率低下。因此，本文首先对 1 200 个正常样本和 1 267 个恶意样本中不同的 N-gram 数进行了统计，观察 N-gram 的个数是否随长度 $N$ 值的增长而呈爆炸式的增长，统计结果如图 4 所示。随着长度的增长，样本中不同的 N-gram 的个数没有呈爆炸式增长，可能的原因是随机组合的指令大多是无意义的指令，能用于程序执行的指令组合只占所有指令组合的小部分。

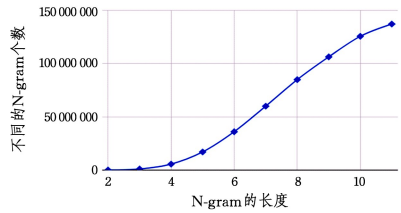


图 4　不同长度下不同的 N-gram 数量

Fig. 4　Number of distinguish N-gram at different length

此外，为了说明 N-gram 分析检测的有效性，本文统计了恶意样本中不同的 N-gram 个数、善意样本中不同的 N-gram 个数以及两类样本重合的 N-gram 种类数，统计结果如表 2 所列。

表 2　恶意样本和善意样本的 N-gram 统计

Table 2　Statistics of N-gram in malicious and benign samples

| N-gram 长度 | 恶意 N-gram 数 | 善意 N-gram 数 | 重合 N-gram 数 |
|---|---|---|---|
| 2 | 32 654 | 41 853 | 31 694 |
| 3 | 413 300 | 917 508 | 343 124 |
| 4 | 1 496 948 | 5 132 118 | 1 030 798 |
| 5 | 3 118 033 | 15 673 153 | 1 789 242 |
| 6 | 4 797 448 | 33 464 685 | 2 259 031 |
| 7 | 6 201 885 | 56 139 781 | 2 343 133 |
| 8 | 7 262 615 | 79 823 452 | 2 151 526 |
| 9 | 8 037 437 | 100 144 931 | 1 852 229 |
| 10 | 8 616 488 | 119 591 633 | 1 564 357 |
| 11 | 9 069 933 | 129 430 910 | 1 330 906 |

同时，图 5 进一步给出了对统计结果的分析值。从图 5

可以看出，当 N-gram 的长度为 2 时，重合的种类数在两者中都占据很大的比例，此时 N-gram 并不能很好地区分恶意应用和善意应用，但是随着 $N$ 值的增加，重合的种类数在善意应用中的比例显著下降，这说明在 $N$ 值足够大（大于或等于 5）时，善意样本只有小部分的 N-gram 与恶意样本相同，因此 N-gram 可作为区分恶意样本和善意样本的特征。

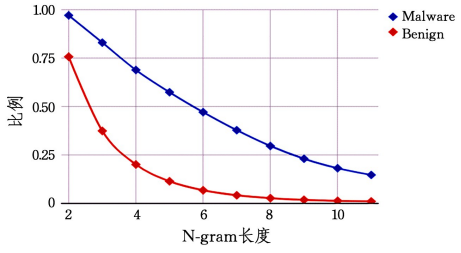


图 5 重合的 N-gram 所占的比例

Fig. 5 Proportion of overlapping N-gram

### 3.2 特征集合选取和深度置信网络

本文首先提取每一个恶意样本的 N-gram 集合，并按照不同 N-gram 出现的频率选取代表恶意应用的恶意 N-gram 特征集 $MSet$，此后利用恶意特征集合向量化测试样本和训练样本。

1）按照频率选取特征集合

对所有的恶意样本进行 N-gram 提取操作，得到全部恶意应用的 N-gram 集合，然后按照不同的 N-gram 出现的频率进行排序，选取前 $k$ 个高频率的 N-gram 作为特征集合，从而得到恶意特征集合为：

$$MSet=\{m_1,m_2,\cdots,m_k\} \tag{2}$$

随后以此恶意特征集合向量化一个 Android 应用。向量化的过程为：首先为样本构造一个 $k$ 维的特征向量 $app\_feacture=[0,0,\cdots,0]$，对于该样本的 N-gram 集合，如果集合中包含恶意特征集合中的元素 $m_i$，那么将向量中下标为 $i$ 的元素置 1，否则置 0，从而得到一个维数为 $k$ 的特征向量。恶意特征集合的大小决定了特征向量的维数，维数过小会降低检测准确率，维数过大则会增加训练和检测的时耗，本文在权衡了准确率和性能之后将 $MSet$ 集合元素的个数设定为 500。

2）深度置信网络模型

在目前基于深度学习理论的模型中，深度置信网络[9]是应用得较为广泛的一种模型。DBN 由多个限制玻尔兹曼机[18]（Restricted Boltzman Machine，RBM）和一层反向传播网络[19]（Back Propagation，BP）组成，与其他的深度神经网络模型相比，DBN 算法能充分挖掘输入向量的特征并且训练速度更快，因此本文使用 DBN 模型用于训练检测。

限制玻尔兹曼机由可见层 $v$ 和隐藏层 $h$ 组成，同一层的节点间互不连接，可见层中的每一个节点和隐藏层中的每一个节点相连。一个典型的 RBM 模型如图 6 所示。

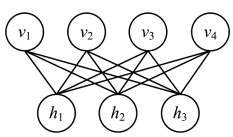


图 6 典型的 RBM 网络

Fig. 6 Typical RBM network

图 6 中 RBM 节点 $v_i$ 到节点 $h_j$ 的权值 $W_{ij}$ 表示节点之间的连接强度，每一个神经元有一个偏置系数 $b_i$（相对于可见神经元）和一个 $c_i$（相对于隐藏神经元）来表示其自身的权重。同时，RBM 神经元的状态是一个二元集合，取值为{0，1}。由此，可以定义一个 RBM 的能量：

$$E(v,h)=-\sum_{i=1}^{N_v}b_i*v_i-\sum_{j=1}^{N_h}c_j*h_j-\sum_{i=1}^{N_v}\sum_{j=1}^{N_h}W_{ij}*v_i*h_j \tag{3}$$

其中，$v_i$ 表示可见神经元的状态，$h_i$ 表示隐藏神经元的状态。在 RBM 中，一个隐藏神经元被激活的概率为：

$$P(h_j|v)=\sigma(c_j+\sum_{i=1}^{N_v}W_{ij}*v_i) \tag{4}$$

其中，$\sigma$ 为 sigmoid 函数。由于 RBM 的可见层和隐藏层之间是双向连接的，可见层的神经元同样可以被隐藏层的神经元激活，被激活的概率为：

$$P(v_i|h)=\sigma(b_i+\sum_{j=1}^{N_h}W_{ij}*h_j) \tag{5}$$

RBM 的训练过程本质上是由可见层构造隐藏层，再由隐藏层重构可见层，反复迭代。经若干次训练之后，隐藏层不仅可以较为精准地表示可见层的特征，还可以还原可见层。

将多个 RBM 串联起来就构成了一个 DBN，其中，上一个 RBM 的隐藏层即为下一个 RBM 的可见层。在训练时，上一个 RBM 必须经过充分训练之后才能训练下一个 RBM。最后，为了将 DBN 用于监督学习，在最后一个 RBM 之后增加一层 BP 网络。一个典型的 DBN 网络的结构如图 7 所示。

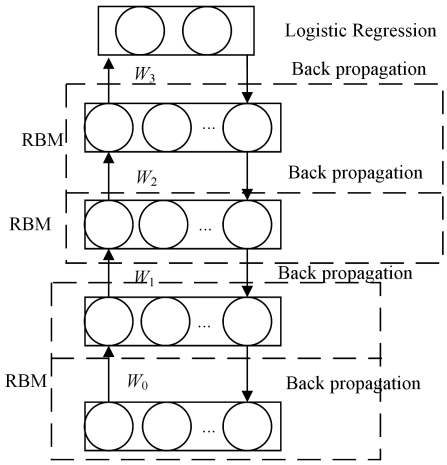


图 7 典型的 DBN 模型

Fig. 7 Typical DBN model

DBN 在训练模型的过程中主要分为两步：

1）分别单独无监督地训练每一层的 RBM，确保特征向量映射到不同的特征空间，并尽量多地保留向量的特征信息。

2）在 DBN 的最后一层 RBM 之后设置一层 BP 网络，以接收最后一层 RBM 的输出。由于之前所有的 RBM 都经过了充分训练，此时各个网络层之间的权值 $W_i$ 都经过了调整，但是每一层的 RBM 网络仅能确保其自身的权值映射达到最优，而不能保证整个 DBN 网络总体最优，因此此时需要用 BP 层的计算结果来反向传播错误信息，开始反向微调网络，以使整个网络达到最优。

DBN 解决了一般多层反向传播网络因随机化的初始权值而陷入局部最优和收敛速度慢的问题。本文的 DBN 使用了 sigmoid 激活函数，DBN 的输入为样本向量化之后的特征

向量，即输入的节点数为 500，输出的节点数为 2，分别对应属于善意或恶意类别的概率。同时 DBN 预学习阶段的学习速率为 0.05，微调阶段的学习速率为 0.2，BP 层采用 Logistic 回归。本文研究了不同的隐藏层结构对恶意检测的影响，实验检验了结构分别为[500,100,2]，[500,150,2]，[500,200,2]，[500,150,100,2]，[500,100,100,2]，[500,150,150,2]，[500,150,100,50,2]时对恶意检测的影响，其中，数组的第一个和最后一个元素表示输入和输出节点层的节点数，其余依次表示隐藏层的节点数。

## 4　实验验证及分析

本文实验选取了 1267 个恶意样本和 1200 个善意样本，善意样本来自 google play，恶意样本来自 genome project[2]。训练集由 90%的恶意样本和 90%的正常样本组成，测试集则由剩下的样本组成。

为了验证本文所提方法的有效性，本文一共设计了 4 组实验。第 1 组实验分析在 N-gram 长度固定时不同的 DBN 隐藏层结构对检测效果的影响；第 2 组实验分析在 DBN 结构固定的情况下不同的 N-gram 长度对检测效果的影响；第 3 组实验将 DBN 模型同传统的机器学习方法进行比较，验证 DBN 模型检测的效果；第 4 组实验将该模型和相关工作的检测模型进行了对比。实验采用精确率(precision)、召回率(recall)、准确率(accuracy)以及 F1 参数(F1-score)来对检测效果进行分析。

$$Precison=\frac{TP}{TP+FP} \tag{6}$$

$$Recall=\frac{TP}{TP+FN} \tag{7}$$

$$Accuracy=\frac{TP+TN}{TP+TN+FP+FN} \tag{8}$$

$$F1\text{-}score=\frac{2*Recall*Precision}{Recall+Precision} \tag{9}$$

*Precision* 表示在预测为正的样本中，正确预测为正样本的概率；*Recall* 则表示在原始正样本中，最后被正确预测为正样本的概率；*Accuracy* 表示整体的预测准确率；*F1-score* 是对精确率和召回率的综合评价，说明了分类模型的稳定程度。

### 4.1　不同 DBN 结构的检测效果

实验测试了在 N-gram 的长度为 5 时不同的 DBN 网络结构对检测效果的影响。实验首先测试了只有一层隐藏层的 DBN 网络结构，此时 DBN 的结构为[500,150,2]，预训练的学习速率设置为 0.05，在微调阶段网络的学习速率设置为 0.2，网络学习的收敛速度快，检测的准确率达到了 91.99%，此时 *F1-score* 为 86.64%，此结果说明模型的分类效果还不稳定。接着实验测试了结构为[500,150,100,2]的网络，保持预训练和微调阶段的学习速率不变，实验结果显示网络收敛的速度较之前慢，但是最后检测的准确率为 93.11%，*F1-score* 为 88.48%，均有所提高。按照同样的思路，实验继续测试了网络结构为[500,150,100,50,2]的 3 层隐藏网络的 DBN 模型，同时测试了层数相同的情况下，隐藏层节点数不同时的检测效果，实验结果如表 3 所列。从表中数据可以看出，隐藏层数的提高使得 DBN 网络学习得更加充分，从而提高了检测准确率，但是网络的收敛速度变慢。另一方面，隐藏层数的提高也有可能使得网络的检测效果有所下降，表 3 中[500,150,100,50,2]结构的网络的检测效果不如[500,150,100,2]结构的网络的检测效果，这可能是因为网络存在过学习(Over Fitting)的情况。从表 3 可以看出，在网络结构为[500,150,100,2]时，DBN 网络的检测准确率最高，达到了 97.8%，同时其 *F1-score* 值也达到了 96.15%，说明此时模型的检测效果稳定。

表 3　不同 DBN 结构的实验结果

Table 3　Experiment results of different DBN structure

(单位：%)

| 网络结构 | *Precision* | *Recall* | *Accuracy* | *F1-score* |
|---|---|---|---|---|
| [500,100,2] | 81.74 | 92.16 | 91.99 | 86.64 |
| [500,150,2] | 83.48 | 94.12 | 93.11 | 88.48 |
| [500,200,2] | 89.72 | 94.12 | 95.31 | 91.87 |
| [500,150,100.2] | 94.34 | 98.04 | 97.80 | 96.15 |
| [500,100,100,2] | 89.81 | 95.10 | 95.59 | 92.30 |
| [500,150,150,2] | 88.99 | 95.10 | 95.32 | 91.94 |
| [500,150,100,50,2] | 89.42 | 91.18 | 94.49 | 90.29 |

### 4.2　不同 N-gram 长度的检测效果

第 3 节中提到，在 N-gram 的长度为 2 时，恶意样本和善意样本重合的 N-gram 种类数在两者中的占比都很大，但随着长度的增加，重合的占比在善意样本中占据的比率的减小速度远大于在恶意样本中占据的比率的减小速度，这在一定程度上说明了 N-gram 的长度达到一定时，N-gram 可以区分恶意样本和善意样本。因此，本文也对不同 N-gram 长度对检测效果的影响进行了实验。实验采用了结构为[500,150,100,2]的 DBN 网络结构，网络预训练阶段的学习速率为 0.05，在微调阶段的学习速率为 0.2。本文对长度为 2～8 的 N-gram 分别进行了实验，检测结果如图 8 所示。从图 8 中可以看出，在 N-gram 的长度为 2 时检测的准确率和 *F1-score* 都不理想，但随着 N-gram 长度的增加，检测的准确率和 *F1-score* 的值都在增加，这符合之前的假设。值得注意的是，在 N-gram 的长度大于 5 之后，检测效果相近，且准确率在长度为 7 时达到了最大值 98.34%，但由于 N-gram 长度的增加，特征集合提取所耗费的资源和时间明显增加。

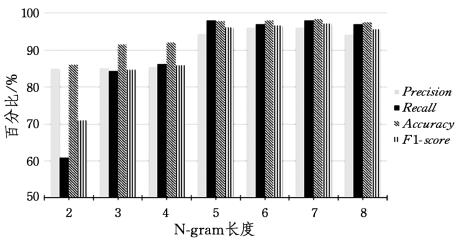


图 8　不同 N-gram 长度的实验结果

Fig. 8　Experiment results of different N-gram length

### 4.3　DBN 与传统机器学习模型的比较

实验比较了在 N-gram 的长度为 5 时 DBN 模型和传统的机器学习算法模型的检测效果。实验采取的 DBN 网络结构为[500,150,100,2]，预训练阶段的学习速率为 0.05，微调阶段的学习速率为 0.2。实验选取 KNN 算法[20]、Decision

Tree 算法[21]以及 SVM 算法[22]与 DBN 模型进行比较，实验结果如图 9 所示。

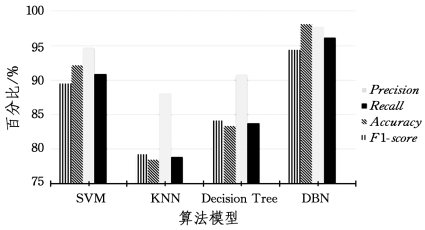


图 9 不同算法模型的检测效果

Fig. 9 Detection results of different algorithm models

从结果可以看出，DBN 检测的准确率以及 *F1-score* 参数都高于其他模型，这说明 DBN 模型的检测效果更优越。

### 4.4 与其他工作的比较

表 4 列出了本文所提出的检测模型和近年来相关工作的对比情况。文献[5]提出的工具检测模型能做到不需要反编译就能提取 Dalvik 指令的形式化特征，但是该工具的预处理部分需要人工分析源代码并提取恶意代码片段，检测效率较低。该模型针对特定类应用的准确率可以达到 100%，但是其实验数据少，结果可能存在误差。文献[6]则利用应用程序的权限特征并结合 K-means 算法进行分析，如果应用程序过度申请敏感权限，模型就会判定该应用是危险的。该模型的优点在于对应用的危险程度进行了分级，但并不能确定应用是否有恶意行为，需要人工进行进一步分析，同时模型所使用的测试数据较少，实验结果也不具备很强的说服力。文献[7]将应用程序的 API 和权限作为整体特征，比较了 KNN 算法以及 SVM 算法的分类结果，使用 SVM 算法对特征进行分类训练，准确率可以达到 90%，而 KNN 算法则达到了 82%。该模型的检测方法无需人工分析应用源码，极大地提高了检测效率，但是选取哪些权限和 API 作为特征需要 Android 安全领域知识，不同的特征选取对结果有较大的影响。文献[9]提出的 Androdect 结合了应用的静态特征以及动态特征，同时利用了大量的测试样本用于测试检测模型，准确率可以达到 94.46%，但是由于引入了动态特征，检测效率也有所降低。

表 4 相关工作对比

Table 4 Related work comparision

| 检测工具 | 选取特征 | 使用算法 | 测试数据 | 检测效果 |
|---|---|---|---|---|
| 基于 Dalvik 指令的代码特征描述[5] | Davilk 指令形式化特征 | 闵可夫斯基距离以及编辑距离 | 8000 个 APK（300 个恶意） | 特定类型应用可以达到 100% |
| 基于 K-means 的权限检测[6] | 申请权限 | K-means 分类 | 40 个 APK（10 个恶意） | 仅能准确预测程序的危险程度 |
| 基于权限和 API 特征结合的检测[7] | 权限以及 API 特征 | SVM 以及 KNN 算法 | 600 个 APK（100 个恶意） | 使用 SVM 训练分类，可以达到 90% |
| 基于多类特征的 Android 恶意检测[8] | 组件、系统调用、函数调用等 | 三层混合系算法 | 3126 个 APK（1126 个恶意样本） | 可以达到 94.46% 的准确率 |
| 基于 N-gram 的检测 | 字节码预处理之后的 N-gram 特征 | DBN 算法模型 | 2467 个 APK（1267 个恶意） | 最高可以达到 98.34% 的准确率 |

本文所提出的基于 N-gram 的检测模型较文献[5]的优点在于不需要人工分析应用程序源码，检测效率高且结果准确，对新的应用也有很好的检测能力。而相较于文献[7-9]，本文模型的优点在于不需要 Android 安全领域的专家知识，减少了人为错误的可能，同时利用 DBN 模型挖掘应用程序更深层次的特征，在保证了检测准确率的同时也确保了检测效率，在网络结构为[500，150，100，2]时准确率最高可以达到 98.34%，高于 Androdect 的 94.46%。

**结束语** 本文提出了利用 N-gram 直接在 Android 应用程序的 Dalvik 字节码上提取特征的技术，并结合深度置信网络(DBN)对应用进行分类训练检测。实验选取了 Google Play 上 1200 个正常的应用样本与 Genome Project 上的 1267 个恶意样本对本文提出的模型进行实验检测。相比于传统的机器学习算法，深度置信网络可以学习到应用程序更加深层次的特征，在 N-gram 的长度为 7 且 DBN 结构为[500，150，100，2]时可以达到 98.34%的准确率。

相比于文献[5-8]，不仅本文所使用的恶意样本的数量有明显增加，本文检测模型的检测效果也更具优势，但仍然缺乏最新的恶意样本，需要继续收集恶意样本源以增强 DBN 的训练。不同的 N-gram 特征提取方法对应用样本的向量化有决定性的影响，即不同的特征提取方法会影响神经网络的输入，进而影响恶意检测结果，本文基于 N-gram 出现的频率选择特征集合，将来还需研究其他可能的方式来提取特征集合，以期提高检测效果。

## 参 考 文 献

[1] TAM K，FEIZOLLAH A，ANUAR N B，et al. The evolution of android malware and android analysis techniques[J]. ACM Computing Surveys (CSUR)，2017，49(4)：76.

[2] ZHOU Y，JIANG X. Dissecting android malware：Characterization and evolution[C]// 2012 IEEE Symposium on Security and Privacy (SP). IEEE，2012：95-109.

[3] QING S H. Research progress on Android security[J]. Journal of Software，2016，27(1)：45-71. (in Chinese)
卿斯汉. Android 安全研究进展[J]. 软件学报，2016，27(1)：45-71.

[4] DESNOS A，GUEGUEN G. Android：From reversing to decompilation[C]// Proceedings of Black Hat Abu Dhabi. 2011：77-101.

[5] LI T，DONG H，YUAN C Y，et al. Description of Android malware feature based on Dalvik instructions[J]. Journal of Computer Research and Development，2014，51(7)：1458-1466. (in Chinese)
李挺，董航，袁春阳，等. 基于 Dalvik 指令的 Android 恶意代码特征描述及验证[J]. 计算机研究与发展，2014，51(7)：1458-1466.

[6] HOU S，DU Y H，LU T L，et al. Research on Android permission detection mechanism based on K-means algorithm[J]. Application Research of Computers，2018，35(4)：1165-1168. (in Chinese)
侯苏，杜彦辉，芦天亮，等. 基于 K-means 算法的 Android 权限检测机制研究[J]. 计算机应用研究，2018，35(4)：1165-1168.

[7] SHAO S D，YU H Q，FAN G S. Detecting Malware by combi-

ning API and Permission Features[J]. Computer Science,2017,44(4):135-139. (in Chinese)
邵舒迪,虞慧群,范贵生. 基于权限和 API 特征结合的 Android 恶意软件检测方法[J]. 计算机科学,2017,44(4):135-139.

[8] YANG H,ZHANG Y Q,HU Y P,et al. A malware behavior detection system of Android applications based on multi-class features[J]. Chinese Journal of Computers,2014,37(1):15-27. (in Chinese)
杨欢,张玉清,胡予濮,等. 基于多类特征的 Android 应用恶意行为检测系统[J]. 计算机学报,2014,37(1):15-27.

[9] HOU S,SAAS A,YE Y,et al. DroidDelver:An Android Malware Detection System Using Deep Belief Network Based on API Call Blocks[C]// International Conference on Web-Age Information Management. Springer International Publishing,2016:54-66.

[10] HINTON G E,OSINDERO S,TEH Y W. A fast learning algorithm for deep belief nets[J]. Neural Computation,2006,18(7):1527-1554.

[11] YANG Z,YANG M. Leakminer:Detect information leakage on android with static taint analysis[C]// 2012 Third World Congress on Software Engineering (WCSE). IEEE,2012:101-104.

[12] GIBLER C,CRUSSELL J,ERICKSON J,et al. AndroidLeaks:Automatically Detecting Potential Privacy Leaks in Android Applications on a Large Scale[C]// Proceedings of International Conference on Trust and Trustworthy Computing. Heidelberg:Springer,2012:291-307.

[13] ARZT S,RASTHOFER S,FRITZ C,et al. Flowdroid:Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps[J]. Acm Sigplan Notices,2014,49(6):259-269.

[14] BODDEN E. Inter-procedural data-flow analysis with ifds/ide and soot[C]// Proceedings of the ACM SIGPLAN International Workshop on State of the Art in Java Program Analysis. ACM,2012:3-8.

[15] OCTEAU D,MCDANIEL P,JHA S,et al. Effective inter-component communication mapping in android with epicc:An essential step towards holistic security analysis[C]// Proceedings of USENIX Security Symposium. Berkeley:USENIX Association,2013:543-558.

[16] SANTOS I,PENYA Y K,DEVESA J,et al. N-grams-based File Signatures for Malware Detection[C]// Proceedings of the 2009 International Conference on Enterprise Information Systems (ICEIS). Heidelberg:Springer,2009:317-320.

[17] APKTOOL. A tool for reverse engineering Android apk files (Version 2.3.4)[EB/OL]. https://ibotpeaches.github.io/Apktool.

[18] HINTON G. A practical guide to training restricted Boltzmann machines[J]. Momentum,2010,9(1):926-947.

[19] SCHMIDHUBER J. Deep learning in neural networks:An overview[J]. Neural Networks,2015,61:85-117.

[20] LIU X M. Anomaly Detection of Malicious Android Application based on K-nearest Neighbor[D]. Beijing:Beijing Jiaotong University,2016. (in Chinese)
刘晓明. 基于 KNN 算法的 Android 应用异常检测技术研究[D]. 北京:北京交通大学,2016.

[21] BARROS R C,BASGALUPP M P,RÉ C,et al. A Survey of Evolutionary Algorithms for Decision-Tree Induction[J]. IEEE Transactions on Systems Man & Cybernetics Part C,2012,42(3):291-312.

[22] GU B,SHENG V S,TAY K Y,et al. Incremental Support Vector Learning for Ordinal Regression[J]. IEEE Transactions on Neural Networks & Learning Systems,2015,26(7):1403-1416.