计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.190700132

基于深度自编码网络的 Android 恶意软件检测方法

孙志强　万　良　丁红卫
贵州大学计算机科学与技术学院　贵阳 550025
贵州大学计算机软件与理论研究所　贵阳 550025
(nightszq@163.com)

摘　要　针对传统 Android 恶意软件检测方法检测率低的问题,文中提出一种基于深度收缩降噪自编码网络(Deep Contractive Denoising Autoencoder Network,DCDAN)的 Android 恶意软件检测方法。首先,逆向分析 APK 文件获取文件中的权限、敏感 API 等 7 类信息,并将其作为特征属性;然后,将特征属性作为深度收缩降噪自编码网络的输入,使用贪婪算法自底向上逐层训练每个收缩降噪自编码网络(Contractive Denoising Autoencoder Network),将训练完成的深度收缩降噪自编码网络用于原始特征的信息抽取,以获取最优的低维表示;最后,使用反向传播算法对获取的低维表示进行训练和分类,实现对 Android 恶意软件的检测。对深度自编码网络的输入数据添加噪声,使得重构的数据具有更强的鲁棒性,同时加入雅克比矩阵作为惩罚项,增强了深度自编码网络的抗扰动能力。实验结果验证了该方法的可行性和高效性。与传统的检测方法相比,该检测方法有效地提高了对恶意软件检测的准确率并降低了误报率。

关键词:Android 恶意软件;深度收缩降噪自编码网络;贪婪算法;反向传播算法;雅克比矩阵
中图法分类号　TP309

Android Malware Detection Method Based on Deep Autoencoder Network

SUN Zhi-qiang,WAN Liang and DING Hong-wei
School of Computer Science and Technology,Guizhou University,Guiyang 550025,China
Institute of Computer Software and Theory,Guizhou University,Guiyang 550025,China

Abstract　To solve the problem of low detection rate of traditional Android malware detection methods,an Android malware detection method based on deep contractive denoising autoencoder network (DCDAN) was proposed. Firstly,the APK file is analyzed in reverse to obtain seven kinds of information in the APK file,such as permissions,sensitive API in the file,which are taken as feature attributes. Then,the feature attributes are taken as the input of the deep contractive denoising autoencoder network,train each contractive denoising autoencoder network is trined layer by layer from bottom to top by using greedy algorithm,and the The deep contractive denoising autoencoder network completed by training is used to extract the information of the original features to obtain the optimal low-dimensional representation. Finally,the back propagation algorithm is used to train and classify the acquired low-dimensional representations to realize the detection of Android malware. Adding noise to the input data of the deep autoencoder network makes the reconstructed data more robust,and adding jacobian matrix as penalty term enhances the anti-disturbance ability of the deep autoencoder network. The experimental results verify the feasibility and high efficiency of this method. Compared with the traditional detection method,the detection method can improve the accuracy of malware detection and reduce the false alarm rate effectively.

Keywords　Android malware,Deep contractive denoising autoencoder network,Greedy algorithm,Back propagation algorithm,Jacobian matrix

1　引言

为了增强市场竞争力,Google 对 Android 系统做出的开放性选择,以及恶意软件的隐蔽性,严重威胁了 Android 系统的安全性。由于系统安全问题的日益突出与反制措施的极不协调[1],对 Android 恶意软件的检测已成为目前的研究热点之一。随着机器学习方法的广泛应用,其也被引入 Android 恶意软件检测中。

相关学者使用传统机器学习方法[2-3]来提高对恶意软件识别和分类的准确率。Sapma 等[4]和 Marios 等[5]通过提取

到稿日期:2019-07-19　返修日期:2019-10-09　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:贵州省科学基金黔科合 LH 字(7634)
This work was supported by the Guizhou Provincial Science Fund LH Word (7634).
通信作者:万良(wanliangtr@163.com)

单一类型的行为特征来检测恶意软件。Wang 等[6]提取了 11 种类型的静态特征来描述软件的行为，并使用机器学习分类器来识别和预测恶意软件。单一的行为特征不能描述软件的全部行为，所以检测效果并不理想。噪声数据会随着特征数据的增多而增多，传统机器学习方法不能很好地挖掘特征数据的深层次信息，从而无法有效地去除干扰信息来获取有用的关键信息，而使用带有干扰信息的数据去训练传统机器学习方法分类器，会导致准确率降低、误报率升高。

深度学习[7-9]是机器学习领域一个新的重要分支，其网络结构和非线性的激活能力非常适合处理高维、海量的数据。Xie 等[10]将卷积神经网络（Convolutional Neural Network，CNN）应用于 Android 恶意软件检测中，实验结果验证了该算法的检测准确率明显高于传统机器学习算法。Luo 等[11]将提取的 API calls，Used permissions 等 5 类信息作为特征并转为灰度图像，将其输入深度置信网络（Deep Belief Network，DBN）中进行恶意软件检测。实验结果显示，与其他浅层模型相比，经过 DBN 降维的数据训练出的模型具有更好的检测能力。Kim 等[12]提出将多模态神经网络（Multimodal Neural Network）作为 Android 恶意软件检测模型，该模型利用静态特征从各方面反映应用程序的属性。实验结果表明，该模型对 Android 恶意软件具有很好的检测效果。

Android 恶意软件所提取的特征数据中存在冗余和噪声等信息，传统机器学习方法通常为浅层的模型结构，不能有效获取特征数据深层次的信息；而深度学习中的深度自编码网络能够近似地还原原始数据，可以很好地获取特征数据更深层的信息，因此本文使用深度自编码网络。但由于良性软件和恶意软件之间存在大量的相似行为，深度自编码网络对其中的微小变化并不敏感，因此结合收缩自编码网络来克服深度自编码网络的缺点；此外，为了增强深度自编码网络去除噪声数据的能力，还结合了降噪自编码网络。综上，文中提出了一种基于深度收缩降噪自编码网络的 Android 恶意软件检测方法，以提高 Android 恶意软件检测的准确率。

2 特征选取

为了获取检测 Android 应用程序所需的特征，本文使用静态分析方法来分析每个应用程序。Android 应用程序的原文件是扩展名为.apk 的压缩文件，使用 Google 提供的开源工具 apktool 来反编译 APK 原文件，获取 AndroidManifest 和 smali 这两个文件，并从这两个文件中提取本文检测所需的特征。

（1）AndroidManifest 文件是应用程序的清单文件，位于整个项目的根目录，描述了全局数据，包含应用程序的配置信息、四大组件信息、权限信息等，为应用程序的运行提供了说明。其中，权限[13]是 Android 系统结构中的重要组成部分，开发人员只有申请了程序运行时所需要的全部权限，应用程序才能被正常安装和使用，文件中通过〈uses-permission〉标签声明；Intent Action 是不同组件之间通信的媒介，文件中通过〈action〉标签声明；Intent Categories 可以反映 Android 中 Activity 的行为，文件中通过〈category〉标签声明；Component 表示 Android 系统的四大组件，即活动、服务、广播接收器和内容提供者，文件中分别通过〈activity〉，〈service〉，〈receiver〉和〈provider〉声明；Hardware 和 Software 表示应用程序的硬件和软件需求，文件中通过〈uses-feature〉声明。Yuan 等[14]的研究显示，恶意软件和良性软件在权限、Intent Action、Intent Categories、Component、Hardware 和 Software 的使用种类和数量上都有较大的差异，因此选取这 6 类信息作为检测的特征。

（2）Smali 是 Android 虚拟机 Dalvik 的反汇编语言，每一个 Smali 文件都对应一个 Java 类，因此其中包含了所有的函数信息。Android 应用程序通过调用这些不同的 API 产生不同的行为，从而完成不同的指令，文件中以 invoke-开头表示函数调用，以此寻找调用函数及相关信息。Aafer 等[15]的研究发现，恶意软件在危险 API 的调用频率和序列上与良性软件有很大的差异，这为恶意软件的检测提供了一定的依据，因此选择 API 调用信息作为检测的又一类特征。

使用正则表达式对各个标签进行匹配，以获取所需特征。本实验一共从以上 7 类信息中获取了 677 个特征，将各种权限、Intent Action 和 Intent Categories 的出现与否量化为 1 和 0，把 API，Component，Hardware 以及 Software 各自的出现次数作为量化结果，使用量化结果构造特征向量，每个应用程序都可以获得一组特征向量作为检测的输入数据。

由于提取的特征数据中各类数值差异较大，不利于检测方法对数据的处理，因此使用最大-最小归一化方法将所有数值映射到[0，1]之间，从而使各个特征数据位于同一数量级，以提高检测方法的收敛速度和精度，如式（1）所示。

$$\hat{x}=\frac{x-x_{\min}}{x_{\max}-x_{\min}} \tag{1}$$

其中，x 为原始数值，$\hat{x}$ 为 x 归一化后的数值，$x_{\min}$ 为当前特征中的最小值，$x_{\max}$ 为当前特征中的最大值。

3 深度收缩降噪自编码网络检测方法

针对传统 Android 恶意软件检测方法检测率低的问题，本文提出了一种新的深度自编码网络来提高 Android 恶意软件的检测准确率。由于恶意软件和良性软件之间具有大量相似的行为特征，且提取的特征中存在噪声数据，因此将能够找出相似特征细微差异的收缩自编码网络与能有效去除噪声的降噪自编码网络相互结合，建立深度收缩降噪自编码网络。

3.1 自编码网络

自编码网络[16]是一种自监督的学习算法，它经过训练尝试将输入数据复制到输出数据，因此可以很好地获取特征数据更深层的信息，更好地区分恶意软件。本文所建立的深度收缩降噪自编码网络由 3 种自编码网络相互组合、堆叠而成。

3.1.1 传统自编码

如图 1 所示，传统自编码网络（Autoencoder Network，AN）是一个最简单的全连接神经网络，只有输入层、隐藏层、输出层 3 层。其中，输入层和输出层的神经元个数相同，这是由自编码网络的输出数据要无限地接近输入数据而决定的；而隐藏层的神经元个数一般少于输入层神经元的个数，具体由所要降低的维度决定。

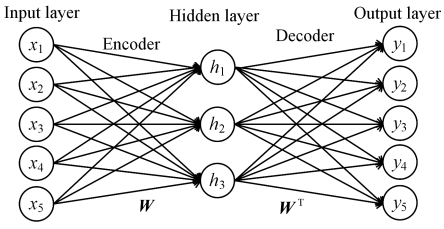

图 1 自编码网络结构

Fig. 1 Autoencoder network structure

AN 是一种数据压缩算法：设原始空间的数据为 $R^{m\times n}$，m 为原始空间中的数据实例，n 为每条实例数据的维度，$x_i\in R^n$ $(i=1,2,\cdots,m)$；将输入数据 x_i 通过非线性函数降低维度输出到隐藏层的过程称为编码过程(Encoder)，使用式(2)得到隐藏层的输出 h_i。

$$h=f(x)=s_f(\boldsymbol{W}x+\boldsymbol{b}_h) \tag{2}$$

$$s(t)=\frac{1}{1+e^{-t}} \tag{3}$$

其中，$\boldsymbol{W}$ 为输入层到隐藏层之间的权重矩阵，$\boldsymbol{b}_h$ 为隐藏层神经元的偏置向量；$s(t)$ 为激活函数，通常取 Sigmoid 函数或恒等函数，本文选择 Sigmoid 非线性激活函数。

从隐藏层的低维特征重构出近似原始数据输出到输出层的过程称为解码过程(Decoder)，使用式(4)得到输出层的重构向量 $\boldsymbol{y}_i$。

$$\boldsymbol{y}=g(h)=s_g(\boldsymbol{W}^{\mathrm{T}}+\boldsymbol{b}_y) \tag{4}$$

其中，$\boldsymbol{W}^{\mathrm{T}}$ 为隐藏层到输出层之间的权重矩阵，$\boldsymbol{b}_y$ 为输出层神经元的偏置向量。

通过反向传播算法微调已有的网络参数，来不断减小重构误差值 J，使输出值尽可能接近输入值，从而学到数据内部的特征。代价函数 J_{AN} 选择均方误差损失函数(MSE)。

$$J_{\mathrm{AN}}=L(x_i,y_i)=\frac{1}{n}\sum_{i=1}^{n}(y_i-x_i)^2 \tag{5}$$

3.1.2 收缩自编码网络

收缩自编码网络[17](Contractive Autoencoder Network，CAN)是 Salah 等于 2011 年提出的一种新正则自编码网络，相比已存在的正则自编码网络具有更好的效果。它主要是对 AN 的编码函数 $f(x)$ 添加了一个正则约束，使其导函数尽可能小，从而保证自编码网络在有扰动的情况下仍然能够很好地训练数据。该过程的具体实现是在原自编码网络代价函数的基础上添加雅克比矩阵。

$$J_{\mathrm{CAN}}=J_{\mathrm{AN}}+\lambda\|J_f\|_F^2 \tag{6}$$

$$\|J_f\|_F^2=\sum_{ij}\left[\frac{\partial f(x)}{\partial x}\right]^2=\sum\{f(x)[1-f(x)]\}^2\sum\boldsymbol{W}_{ij}^2 \tag{7}$$

其中，λ 是权衡代价函数和约束项之间的比例系数。雅克比矩阵主要表现为一个多变数向量函数的最佳线性逼近，因此可以抵抗数据在各方向上的扰动，获取应用程序相似行为的细微差异，从而增强自编码网络的特征提取能力。

3.1.3 降噪自编码网络

降噪自编码网络(Denoising Autoencoder Network，DAN)是 Pascal 等[18]提出的，可以有效去除噪声，还原原始数据，使自编码网络学习到的特征数据具有更强的鲁棒性。DAN 的网络结构如图 2 所示。

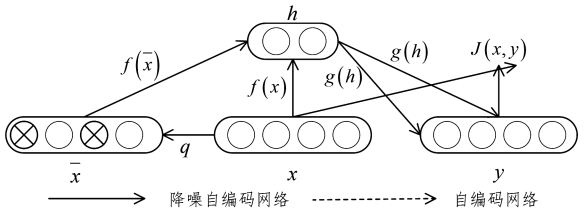

图 2 降噪自编码网络的结构

Fig. 2 Structure of denoising autoencoder network

降噪自编码网络主要是在自编码网络的基础上引入了一个数据损坏的过程，即原始数据 x，使用随机映射函数 q 添加一定比例的噪声数据，得到受损的输入数据 $\bar{x}$；然后，把 $\bar{x}$ 输入式(2)中进行编码获得 h，再把 h 输入式(4)中，根据未受干扰的数据估计受干扰数据的原始形式 y；最后，通过梯度下降算法不断优化 x 和 y 之间的重构误差 J，使得 y 无限接近于未添加噪声的原始数据 x，这就迫使隐藏层学到的数据特征具有更强的鲁棒性，避免了输出数据直接复制输入数据，从而增强了自编码网络的特征表达能力。

3.2 深度收缩降噪自编码网络

DCDAN 是一种新的深度自编码网络，如图 3 所示。首先，在自编码网络的输入层和隐藏层之间加入降噪自编码网络的数据损坏过程，对数据添加噪声，再将自编码网络的代价函数替换为收缩自编码网络的代价函数，从而构建了一个 CDAN；然后，将多个 CDAN 堆叠在一起，使前一个 CDAN 的输出为后一个 CDAN 的输入；最后，在最后一个 CDAN 后添加一层 BP 神经网络作为分类层，这就组成了 DCDAN。

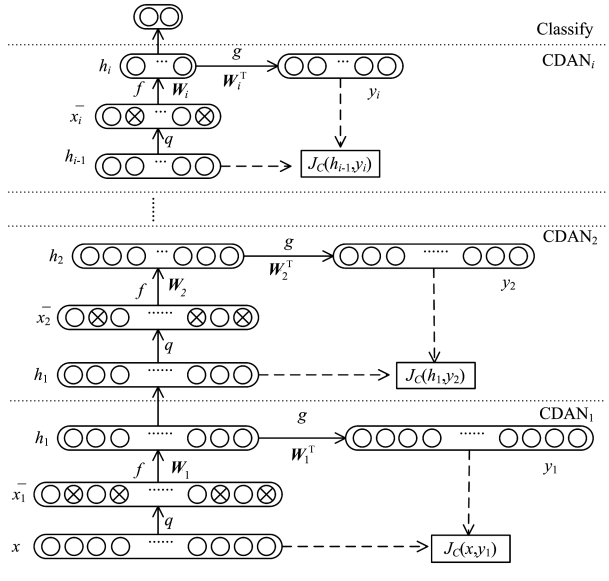

图 3 深度收缩降噪自编码网络结构

Fig. 3 Deep contractive denoising autoencoder network structure

DCDAN 的工作原理是通过逐层减少隐藏层中的神经元个数来获取数据的深层信息，去除干扰信息和非本质信息，以较少的低维特征来表示原始的高维特征，再使用反向传播算法对获取的低维特征进行分类。DCDAN 可以更好地处理微弱的信息，最大化地区分相似行为的不同点，使学习到的特征数据具有局部不变性和更强的鲁棒性。

随着网络层数的增加，传统的训练方法会出现梯度消失问题，即优化网络参数时前面隐藏层的学习速率低于后面隐藏层的学习速率，导致网络参数无法更新，因此将 DCDAN 的训练分为两步：预训练和微调。首先，将每一个 CDAN 看作单独的一部分，逐个进行训练，将前一个 CDAN 训练好后获得的最优输出作为后一个 CDAN 的输入，以此完成对每个 CDAN 的训练。通过这种无监督的逐层训练也就完成了对堆叠 CDAN 的预训练。之后将这些堆叠 CDAN 展开，得到 DCDAN，并对堆叠 CDAN 的参数进行整体有监督的微调，从而使 DCDAN 达到最优，这样就解决了梯度消失的问题，可以获取最优的参数。

(1)无监督的预训练

主要使用贪婪算法逐层训练每一个 CDAN。

1)随机初始化权重 $\boldsymbol{W}_i$ 和偏置 $\boldsymbol{b}_i$，设置隐藏层层数 N_i、惩罚项系数 λ 和噪声系数 γ，其中 i 为收缩降噪自编码网络的堆叠数；

2)将本文提取的特征数据 x 传入输入层；

3)对输入数据 x 添加一定比例的噪声，形成带有损伤的新数据 $\bar{x}$；

4)将新数据 $\bar{x}$ 输入未训练的 CDAN 中，再使用式(2)进行编码，获取其隐藏层神经元的输出 h；

5)使用式(4)对隐藏层的输出 h 进行解码，以获得重构数据 y；

6)使用式(6)的代价函数 $J_{\mathrm{CAN}}(x,y)$ 不断优化重构误差 J，从而更新 $\boldsymbol{W}_i$ 和 $\boldsymbol{b}_i$；

7)获取训练好的 CDAN 中的 W_i 和 b_i 并对其进行保存；

8)将新数据 $\bar{x}$ 再次输入刚训练好的 CDAN 中，经过式(2)编码获得隐藏层的输出 $\bar{h}$；

9)将 $\bar{h}$ 作为下一个 CDAN 的输入，重复步骤 3)－9)，依次完成每个 CDAN 的训练。

(2)有监督的微调

通过无监督的预训练后，得到了堆叠的训练好的 CDAN，将其展开后得到如图 4 所示的对称 DCDAN。

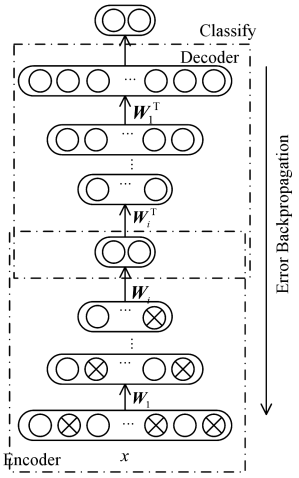

图 4 有监督的微调

Fig. 4 Supervised fine tuning

其中，前 i 层为编码器，后 $i+1$ 到 $2i$ 层为解码器，第 $2i+1$ 层为添加的分类层，用于接收最后一个 CDAN 输出的低维表示的特征数据，完成对恶意软件的分类。由于无监督的预训练只能保证每个 CDAN 的参数达到最优，并不能使整个 DCDAN 的参数达到最优，因此需要使用反向传播算法对 CDAN 的参数进行有监督的微调，同时反向传播算法还具有分类的功能，具体过程如下。

1)将带标签的本文特征数据 x 输入展开后的 DCDAN 中，使用前向传播算法即式(8)依次计算各层神经元的激活值。

$$\bar{y}=s(x\boldsymbol{W}_i+\boldsymbol{b}_i) \tag{8}$$

其中，$\boldsymbol{W}_i$ 为权重，$\boldsymbol{b}_i$ 为偏置，其值为无监督的预训练过程中存储的每个训练好的 CDAN 的 $\boldsymbol{W}_i$ 和 $\boldsymbol{b}_i$；s 为式(3)的 Sigmoid 激活函数。

2)误差反向传播：将真实值 $\hat{y}$ 和激活值 $\bar{y}$ 输入式(9)，以获取损失函数的值。

$$E(\theta)=\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i-\bar{y}_i)^2 \tag{9}$$

其中，θ 表示参数集合，n 表示输出数据的维数。

3)根据损失函数的值来计算输出层的误差项和隐藏层的误差项，r 为前一层的一个神经元，u 为中间层的一个神经元，v 为后一层的一个神经元。

输出层的误差项：权重的误差计算公式如式(10)所示，偏置的误差计算公式如式(11)所示。

$$\Delta_{(k)}\boldsymbol{W}_{uv}=\frac{\partial E}{\partial \boldsymbol{W}_{uv}}=\frac{\partial net_2}{\partial \boldsymbol{W}_{uv}}\frac{\partial \bar{y}}{\partial net_2}\frac{\partial E}{\partial \bar{y}} \tag{10}$$

$$\Delta_{(k)}\boldsymbol{b}_v=\frac{\partial E}{\partial \boldsymbol{b}_v}=\frac{\partial net_2}{\partial \boldsymbol{b}_v}\frac{\partial \bar{y}}{\partial net_2}\frac{\partial E}{\partial \bar{y}} \tag{11}$$

隐藏层的误差项：权重的误差计算公式如式(12)所示，偏置的误差计算公式为式(13)所示。

$$\Delta_{(k)}\boldsymbol{W}_{ru}=\frac{\partial E}{\partial \boldsymbol{W}_{ru}}=\frac{\partial net_1}{\partial \boldsymbol{W}_{ru}}\frac{\partial h}{\partial net_1}\frac{\partial net_2}{\partial h}\frac{\partial \bar{y}}{\partial net_2}\frac{\partial E}{\partial \bar{y}} \tag{12}$$

$$\Delta_{(k)}\boldsymbol{b}_u=\frac{\partial E}{\partial \boldsymbol{b}_u}=\frac{\partial net_1}{\partial \boldsymbol{b}_u}\frac{\partial h}{\partial net_1}\frac{\partial net_2}{\partial h}\frac{\partial \bar{y}}{\partial net_2}\frac{\partial E}{\partial \bar{y}} \tag{13}$$

其中：

$$net_1=\boldsymbol{W}_{ru}x+\boldsymbol{b}_u \tag{14}$$

$$h=s(net_1) \tag{15}$$

$$net_2=\boldsymbol{W}_{uv}h+\boldsymbol{b}_v \tag{16}$$

4)更新权重和偏置。

输出层参数更新：权重更新如式(17)所示，偏置更新如式(18)所示。

$$\boldsymbol{W}_{uv}^{(k)}=\boldsymbol{W}_{uv}^{(k-1)}-\eta\Delta_{(k)}\boldsymbol{W}_{uv}=\boldsymbol{W}_{uv}^{(k-1)}-\eta\frac{\partial E}{\partial \boldsymbol{W}_{uv}} \tag{17}$$

$$\boldsymbol{b}_v^{(k)}=\boldsymbol{b}_v^{(k-1)}-\eta\frac{\partial E}{\partial \boldsymbol{b}_v} \tag{18}$$

隐藏层参数更新，权重更新如式(19)所示，偏置更新如式(20)所示。

$$\boldsymbol{W}_{ru}^{(k)}=\boldsymbol{W}_{ru}^{(k-1)}-\eta\Delta_{(k)}\boldsymbol{W}_{ru}=\boldsymbol{W}_{ru}^{(k-1)}-\eta\frac{\partial E}{\partial \boldsymbol{W}_{ru}} \tag{19}$$

$$\boldsymbol{b}_u^{(k)}=\boldsymbol{b}_u^{(k-1)}-\eta\frac{\partial E}{\partial \boldsymbol{b}_u} \tag{20}$$

其中，η 表示学习率，$k=1,2,\cdots,t$ 表示更新次数。

5)激活值达到预期目标时停止，否则重复步骤 1)－5)，直至获得的激活值达到预期的目标，此时的参数为 DCDAN 的最优参数。

3.3 Android 恶意软件检测过程

使用 DCDAN 对 Android 恶意软件进行检测的过程如图 5 所示。

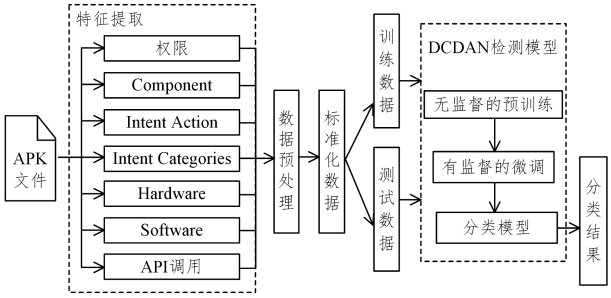

图 5 Android 恶意软件的检测过程

Fig. 5 Detection process of Android malware

首先，使用 apktool 工具对收集的 APK 原文件进行批量反编译，获取每个 APK 原文件对应的 AndroidMainfest 文件和 smali 文件。其次，使用正则表达式编写所需特征的匹配规则并与文件中数据进行匹配，以获取所需特征。每个 APK 原文件都可以获得一组特征，将获取的所有组的特征进行最大-最小归一化的预处理，以获取标准化数据，再从标准化数据中随机选取一大部分作为训练数据，另一小部分作为测试数据。然后，使用训练数据完成深度收缩降噪自编码网络的训练。最后，将测试数据输入训练好的深度收缩降噪自编码网络中，以获得每条数据的预测结果。

4 实验

实验的硬件环境为 Inter Core i5-3230M CPU 2.60 GHz，8 GB RAM；软件环境为 Windows10，python3.6，Java。

4.1 实验数据

为了验证深度收缩降噪自编码网络检测 Android 恶意软件的能力，选取了 2500 个恶意软件和 2500 个良性软件作为训练集。其中，恶意软件均来自于 DREBIN 数据集[19-20]，该数据集包含了 179 类恶意软件，共 5560 个恶意样本，随机选取其中的 2500 个用于实验；而 2500 个良性软件是使用网络爬虫工具从 Google Play 上随机获取的，虽然 Google 部署了 Android 恶意软件检测服务 Bouncer，使得 Google Play 中恶意软件的感染率极低，但为了保证实验的准确性，使用 VirusTotal 网站提供的恶意检测服务对获取的所有良性样本进行检测，去除良性样本中可能隐藏的恶意软件。另外，重新随机获取良性、恶意软件共 500 个作为测试集。

4.2 评价指标

为了客观地评价本文检测方法的好坏，将准确率(Accuracy，ACC)、良性召回率(Benign Recall，BR)、恶意召回率(Malicious Recall，MR)、误报率(False Alarm Rate，FA)、综合评价指标(F-Measure，F_1)作为评估指标，表 1 列出了 4 个基本评价指标。

表 1 基本评价指标

Table 1 Basic evaluation index

实际分类	预测分类		总计
	恶意软件	良性软件	
恶意软件	TN	FP	N
良性软件	FN	TP	P

基于表 1 的 4 个基本指标，定义以下 5 个度量指标。

(1)ACC：一共被正确识别出来的恶意软件和良性软件个数。

$$ACC=\frac{TP+TN}{TP+FN+TN+FP} \tag{21}$$

(2)BR：被正确识别出来的良性软件个数在所有良性软件个数中的占比。

$$BR=\frac{TP}{TP+FN} \tag{22}$$

(3)MR：被正确识别出来的恶意软件个数在所有恶意软件个数中的占比。

$$MR=\frac{TN}{TN+FP} \tag{23}$$

(4)FA：被误报为良性软件的恶意软件个数占所有恶意软件个数的比例。

$$FA=\frac{FP}{FP+TN} \tag{24}$$

(5)F_1：精确率(Precision)和召回率(Recall)的加权调和平均值。

$$F_1=\frac{2\times(Precision\times Recall)}{Precision+Recall} \tag{25}$$

良性软件的准确率为：

$$Precision_{良}=\frac{TP}{TP+FP} \tag{26}$$

恶意软件的准确率为：

$$Precision_{恶}=\frac{TN}{TN+FN} \tag{27}$$

4.3 深度收缩降噪自编码网络的结构分析

深度收缩降噪自编码网络通过叠加收缩降噪自编码网络，可以增加深度收缩降噪自编码网络中隐藏层的个数。虽然个数越多，越可以表示更高层的含义，但这并不意味着编码的效果就越好。因此，需要研究训练隐藏层的个数以及如何设置隐藏层中神经元的个数，从而获得良好的分类效果。目前，针对这个问题并没有一个准确的理论方法，只能在已有的研究上进行大量的实验，对比不同深度、降维维度对分类效果的影响。因此，本文设置了 7 种 DCDAN，网络深度从 2 层增加到 6 层(这里讨论的是 DCDAN 的深度，由于最后一层是分类层，所以不计入网络深度中)，分类结果如表 2 所列。

表 2 不同深度、维度的 DCDAN 分类结果

Table 2 DCDAN classification results of different depths and dimensions

网络结构	深度	ACC/%
677-30	2	95.4
677-320-30	3	96.0
677-320-160-30	4	96.6
677-320-160-80-20	5	97.4
677-320-160-80-30	5	97.8
677-320-160-80-40	5	97.0
677-320-160-80-30-10	6	96.4

从表 2 可知，具有 5 层深度且压缩提取 30 维新特征的 DCDAN 具有良好的分类效果。一般情况下，随着网络深度的不断增加，深度自编码网络越能更好地去除特征数据的冗余和噪声，其抽象概括特征数据的能力也增强。但当深度增加超过了概括数据的临界点时，就会使得提取的新数据特征过于抽象，丢失大量的本质信息，增大训练误差，从而无法有效地重构出原始数据。而当网络深度较浅时，由于压缩过快，不能对特征数据进行有效的分析，同样会损失部分重要信息，不利于挖掘本质特性。同理，最后压缩的维度若过小，会使数据笼统化，从而丢失大量的关键信息；若过大，则特征数据的概括性不足，导致其中仍包含大量的原始冗余和噪声，不利于对特征数据进行分析与分类。

使用训练集中的数据对 5 层深度的 DCDAN 进行训练和预测，得到如图 6 所示的降维效果、图 7 所示的训练和验证准确率、图 8 所示的训练和验证损失值。其中，从训练集中随机选取了 20%作为验证集。从图 6 中可以看出，DCDAN 可以很好地保留有用的原始信息，实现了数据的有效分离；图 7 中，准确率稳步上升，达到了一个较高的值；图 8 中，损失函数收敛迅速，且稳定于一个较低的值。因此，本文将 DCDAN 的网络结构设置为 677-320-160-80-30，对于其他参数，通过大量的实验不断进行分析、调优，最终确定了如表 3 所列的参数。

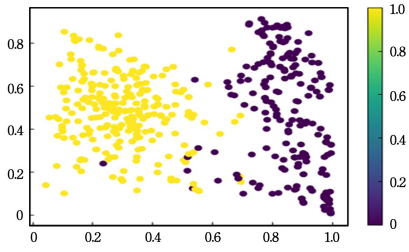

图 6 降维效果图

Fig. 6 Dimensionality reduction

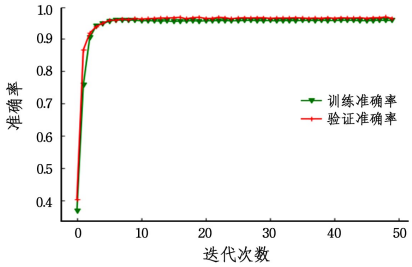

图 7 准确率曲线

Fig. 7 Accuracy curve

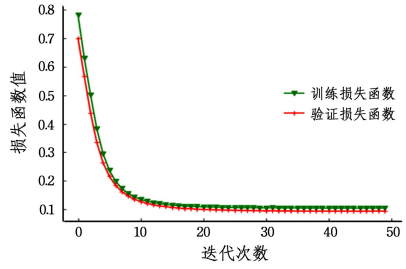

图 8 损失值曲线

Fig. 8 Loss value curve

表 3 DCDAN 的参数设置

Table 3 Parameter setting of DCDAN

参数	数值	参数	数值
网络结构	677-320-160-80-30	Dropout	0.2
惩罚项系数 λ	0.04	Pre-training epoch	100
噪声系数 γ	0.001	Fine-tuning epoch	50
学习率	0.001		

4.4 对比实验

为了验证本文提取的特征描述恶意软件行为的效果以及提出的 DCDAN 检测方法的有效性，设计了下面 3 组不同的实验。

(1)不同特征的比较

权限和 API 调用是研究人员常使用的静态特征，它们都可以在一定程度上反映应用程序的行为，所以可以单独作为特征使用。本文对比了权限、API 调用和所提的 7 类结合特征，分类结果如图 9 所示，误报率如图 10 所示。

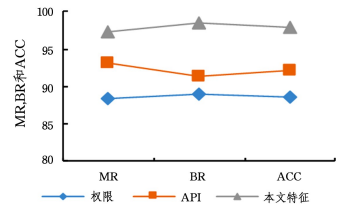

图 9 不同特征的对比

Fig. 9 Accuracy of different features

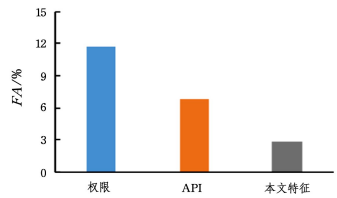

图 10 不同特征的误报率

Fig. 10 False positive rate of different characteristics

从图中可以看出，本文所选取的特征可以更好地描述软件的行为，对恶意软件和良性软件的检测都有较高的召回率，并且降低了误报率；而权限特征和 API 特征的单独使用，都只描述了应用程序行为的一个方面，破坏了特征间的内在联系，不能全面地反映应用程序的行为特征，致使无法有效地对大量相似的行为进行区分，从而使得准确率较低、误报率较高。

(2)学习能力的比较

为了验证 DCDAN 对特征的学习能力，选择不同数量以及不同比例的恶意软件和良性软件进行对比。详细的设置和分类结果如图 11 和图 12 所示。

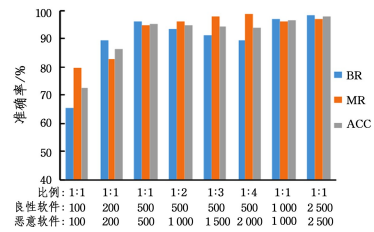

图 11 不同数量及比例对学习能力的影响

Fig. 11 Impact of different numbers and proportions on learning ability

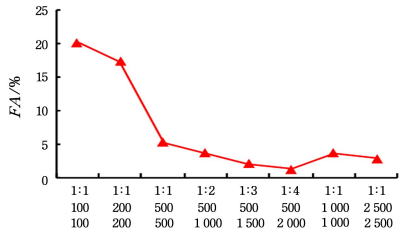

图 12 不同数量及比例的误报率

Fig. 12 False positives rate of different numbers and proportions

实验结果表明，随着恶意软件和良性软件数量的不断增加，DCDAN 的学习能力越来越强，可以更好地获取特征的本质信息。当恶意软件和良性软件的比例为 1∶1 时，随着软件数量的不断增加，恶意软件召回率和良性软件召回率相差不大，且都平稳增长，整体准确率也不断提升，误报率平稳降低，所以可以通过训练更多的数据来提高准确率；当恶意软件和良性软件的数量不相等时，逐渐增加恶意软件的数量，DCDAN 可以学习到更多的恶意软件行为特征，使得恶意召回率较高，误报率较低，由于对良性软件的行为学习不足，因此良性召回率并不是很高，但整体可以保持较高的准确率。

(3)不同方法的比较

为了证明本文提出的 DCDAN 检测方法具有更好的识别和分类能力，与 DBN，CNN 和多种机器学习方法的分类结果进行比较，始终使用文中提取的 500 测试集进行测试，结果如表 4 所列。

表 4 不同分类算法的准确率对比

Table 4 Accuracy comparison of different classification algorithms

(单位：%)

算法	恶意软件		良性软件		ACC
	MR	F_1	BR	F_1	
DBN	95.9	96.7	97.6	96.8	96.8
CNN	97.6	97.1	96.8	97.2	97.2
J48	86.7	90.1	94.4	91.0	90.6
KNN	99.6	91.1	81.3	89.5	90.4
Logistic	81.9	88.6	97.2	90.4	89.6
Naïve Bayes	64.5	76.7	96.4	83.4	80.6
Random Forest	98.4	96.1	93.7	95.9	96.0
SVM	98.4	95.3	92.1	95.1	95.2
DCDAN	97.2	97.8	98.4	97.8	97.8

从表 4 可以看出，DCDAN 检测方法对恶意软件的检测效果优于其他检测方法。其中，Naïve Bayes 的检测效果最差，准确率仅有 80.6%；而检测效果最好的 CNN 的准确率达到了 97.2%，但仍比 DCDAN 低 0.4%。因此，该 DCDAN 方法具有良好的检测效果。

结束语 针对传统的检测方法对恶意软件检测率低的问题，本文提出了一种基于深度收缩降噪自编码网络的 Android 恶意软件检测方法。首先，选取权限、API 调用和 Intent Action 等多类特征从各层面描述 Android 恶意软件的行为；然后，通过收缩降噪自编码网络无监督自主的学习特征规律，去除噪声冗余以提取深层次的抽象特征，其中，加入的降噪自编码网络有效地增强了降维后数据的鲁棒性，加入的收缩自编码网络增大了相似行为间的细微差异；最后，将各个已训练好的收缩降噪自编码网络的参数作为深度收缩降噪自编码网络参数的初始值，使用反向传播算法进行微调并对降维后的数据进行分类。实验结果表明，与传统的检测方法相比，本文提出的 DCDAN 检测方法具有更高的检测率，并且非常适合提取特征数据的深层信息，找出恶意软件和良性软件之间相似行为的细微差异，这为当前的 Android 恶意软件检测研究提供了一种新思路。本实验仅提取静态特征进行了分析，未来计划将软件放入虚拟机中运行来获取动态特征，从静态和动态两大方面描述软件的行为，从而更好地识别 Android 恶意软件。

参 考 文 献

[1] QING S H. Research Progress on Android Security [J]. Journal of Software, 2016, 27(1): 45-71.

[2] VINOD P, AKKA Z, MAURO C. A machine learning based approach to detect malicious android apps using discriminant system calls[J]. Future Generation Computer Systems, 2019, 94: 333-350.

[3] HE G F, XU B F, ZHU H T. AppFA: A Novel Approach to Detect Malicious Android Applications on the Network[J]. Security and Communication Networks, 2018, 2018(4): 1-15.

[4] SAPNA M, KIRAN K. Malicious Application Detection and Classification System for Android Mobiles [J]. International Journal of Ambient Computing and Intelligence, 2018, 9: 95-114.

[5] MARIOS A, NICOLA D, ANGELO S. Analysis and Evaluation of SafeDroid v2. 0, a Framework for Detecting Malicious Android Applications[J]. Security and Communication Networks, 2018(1): 1-15.

[6] WANG W, LI Y Y, WANG X, et al. Detecting Android malicious apps and categorizing benign apps with ensemble of classifiers[J]. Future Generation Computer Systems, 2018, 78(3): 987-994.

[7] JUERGEN S. Deep Learning in Neural Networks: An Overview [J]. Neural Networks, 2015, 61: 85-117.

[8] PRASHANT K, PANKAJ K. A Novel Approach for Detecting Malware in Android Applications Using Deep Learning[C]// 2018 Eleventh International Conference on Contemporary Computing(IC3). IEEE Computer Society, 2018, 1: 1-4.

[9] LI D F, WANG Z G, XUE Y B. Fine-grained Android Malware Detection based on Deep Learning[C]// 2018 IEEE Conference on Communications and Network Security (CNS). Beijing: IEEE, 2018: 1-2.

(文献[10-20]见文章首页二维码)

SUN Zhi-qiang, born in 1995, postgraduate, is a member of China Computer Federation. His main research interests include malware detection, information security and deep learning.

WAN Liang, born in 1974, Ph.D, professor, is a member of China Computer Federation. His main research interests include information security, network security and deep learning.