doi:10.11959/j.issn.1000-436x.2017259

基于图结构的恶意代码同源性分析

赵炳麟，孟曦，韩金，王婧，刘福东
（数学工程与先进计算国家重点实验室，河南 郑州 450002）

摘 要：恶意代码检测和同源性分析一直是恶意代码分析领域的研究热点。从恶意代码提取的 API 调用图，能够有效表示恶意代码的行为信息，但由于求解子图同构问题的算法复杂度较高，使基于图结构特征的恶意代码分析效率较低。为此，提出了利用卷积神经网络对恶意代码 API 调用图进行处理的方法。通过选择关键节点，以关键节点邻域构建感知野，使图结构数据转换为卷积神经网络能够处理的结构。通过对 8 个家族的恶意样本进行学习和测试，实验结果表明，恶意代码同源性分析的准确率达到 93%，并且针对恶意代码检测的准确率达到 96%。
关键词：恶意代码；同源性分析；API 调用图；卷积神经网络
中图分类号：TP302 **文献标识码**：A

Homology analysis of malware based on graph

ZHAO Bing-lin, MENG Xi, HAN Jin, WANG Jing, LIU Fu-dong
(State Key Laboratory of Mathematical Engineering & Advanced Computing, Zhengzhou 450002, China)

Abstract: Malware detection and homology analysis has been the hotspot of malware analysis. API call graph of malware can represent the behavior of it. Because of the subgraph isomorphism algorithm has high complexity, the analysis of malware based on the graph structure with low efficiency. Therefore, this studies a homology analysis method of API graph of malware that use convolutional neural network. By selecting the key nodes, and construct neighborhood receptive field, the convolution neural network can handle graph structure data. Experimental results on 8 real-world malware family, shows that the accuracy rate of homology malware analysis achieves 93%, and the accuracy rate of the detection of malicious code to 96%.
Key words: malware, homology analysis, API call graph, convolutional neural network

1 引言

根据 360 公司“2016 年互联网安全报告”，2016 年共截获 PC 端新增恶意程序样本 1.9 亿个[1]。面对海量新出现恶意代码，基于特征码和签名的传统恶意代码分析方法已经不能满足恶意代码的检测需求，启发式方法、云检测技术以及主动防御技术被提出，并已经应用于恶意代码关联分析和同源分析。

若 2 个恶意代码由同一恶意代码采用代码复用的方式演变而来，或行为具有相似性，而出现存在先后的关系，则称它们同源。随着攻击向高级、持续（如 APT，advanced persistent threat）等方向发展，恶意代码攻击场景更加复杂，为了躲避检测，恶意代码不断采用多态和变形等方式。因此，发现样本中的同源关系对攻击组织溯源、运行环境还原以及攻击防范具有重要的作用。

当前恶意代码同源性分析，主要利用动态跟踪或静态分析获取恶意代码的特征信息，如指令序列[2]、API（application programming interface）调用序列[3]或图结构特征[4]等。通过对恶意代码的特征进行学习，建立不同类别恶意代码的特征模型，通过计算待检测恶意代码针对不同类别的同源度，指导恶意代码的同源性判定。序列信息比图结构信息更容易获得，因此大多数研究者基于指令序列或 API 序列进行研究。但相比图结构信息，序列信息会丢失恶意代码执行过程中的某些空间特征。另一方面，针对图结构数据的匹配和分析，在计算同源性的过程中需要涉及子图同构问题[5]，而子图同构问

收稿日期：2011-09-10

题是 NP 完全问题，随着图规模提高，计算复杂度呈指数级增长，效率得不到有效保证。

为了解决恶意代码同源性分析过程中图结构数据计算复杂度较高的问题，本文在提取恶意代码 API 调用图的基础上，提出了一种基于图结构的恶意代码同源性分析方法。该方法以 API 调用图作为研究对象，利用卷积神经网络对图结构数据进行处理。为了使不同大小的 API 调用图能够适配卷积神经网络的输入，提出了一个关键节点选择算法。该算法通过计算节点在图中的重要性，选取关键节点，然后以关键节点的邻域，构建感知野。由于卷积神经网络良好的局部性，有效提高了图匹配的计算效率，对大量恶意代码进行了同源性分析，取得较好效果。

本文的主要贡献包含以下几个方面。

1) 提出了恶意代码同源性分析方法，在恶意代码同源性分析中，采用卷积神经网络处理图数据结构，为恶意代码同源性分析提供了一个新的思路。

2) 通过对静态分析提取的 API 调用图进行改造，使其能够作为卷积神经网络的输入。并提出了针对 API 调用图的关键节点选择和邻域节点遍历算法，建立节点的感知野。

3) 通过在公开的恶意代码数据集上进行大量实验，所提方法准确率达到 96%。

2　相关研究

卷积神经网络作为深度神经网络中广泛应用的结构，其局部连接、权值共享以及池化操作等特性使其能够有效降低网络的复杂度，减少训练参数的数目以及具有强顽健性和容错能力，且易于优化和训练。因此，卷积神经网络在图像识别、序列检索、目标识别等方面有较快发展。由于卷积神经网络中感知野的存在，其具有很好的局部性，在子图同构和图相似性度量的过程中也具有较大的潜力，目前也有一些基于卷积神经网络的图结构数据分析方法[6~9]，并取得一定成果。

恶意代码同源性分析主要采用静态或动态的方式获取恶意代码的特征信息，包括序列信息以及图结构信息，通过计算特征的相似性，得到恶意代码同源性的判断。

采用序列信息进行相似性判定，主要以恶意代码提取的序列特征作为对象进行分析。Youngjoon 等[10]提出了基于 API 序列的恶意代码检测方法。该方法在沙箱环境动态运行恶意代码，捕获其运行过程中的 API 序列，利用生物基因序列检测工具 ClustalX[11]，对 API 序列进行相似性分析，最后得到恶意代码的同源性判定。Lee 等[12]提出 API 序列与 n-gram 结合的方法来分析恶意代码的同源性并识别恶意代码的变异。该方法通过虚拟环境 Cuckoo 等[13]动态获取恶意代码执行过程中的 API，并对 API 序列进行 n-gram 转换，然后根据 n-gram 的相似性，计算恶意代码之间的同源性，发现恶意代码变种。Cesare 等[14]通过提取恶意代码的控制流结构，并将控制流结构转换为对应的二维表示，有效降低相似性比较和搜索的复杂度。基于序列的恶意代码同源性分析主要通过针对提取得到的序列，计算序列的相似程度。例如，最长公共子序列（LCS，longest common subsequence）算法，或采用 n-gram 算法，将序列转化成 n-gram 序列进行相似性判断。现有基于序列的恶意代码分析方法对序列提取要求较高，而序列往往是在图结构基础上进行遍历和拆分得到的，在对比过程中会损失一定逻辑关系和空间属性。

文献[15]通过动态捕获恶意软件行为的方式构造行为的系统调用图进行分类，确定代码相似性。而 Kinable 等通过静态分析恶意代码的系统调用图，采用图匹配的方式比较恶意代码的相似性，进行家族分类[16]。文献[17]通过提取函数调用关系，构造函数调用图，将函数调用图转换成矢量表示，有效降低了计算复杂度。杨帆等[18]通过静态分析抽象函数特征，采用图的编辑距离判定恶意代码的相似性。刘星等[19]通过计算函数调用图的相似性距离对恶意代码同源性进行分析。

采用图结构表示恶意代码的信息能够较好地保留恶意代码的整体性特征，但是由于子图同构问题是 NP 完全问题，所以针对图进行相应的同源性分析，其复杂度往往较大。一般解决方案从优化图结构入手，将图结构进行精简。但这种情况没有从根本上解决图的相似性匹配问题，所以本文以图结构作为恶意代码同源性分析的切入点，进行深入研究。

3　基于图结构的恶意代码同源性分析模型

3.1　相关定义

定义 1　（API 调用图）一个 API 调用图 G 是一个有向图，它包含 2 个元素，$G=(V, E)$，其中，V 为一个有限顶点集 $V=\{v_1, v_3, \cdots, v_N\}$，且每个顶点都

对应一个 API 调用，$E=\{\langle v_i,v_j\rangle \mid v_i,v_j\in V\}$ 为一个有向边的集合，即表示每个顶点的前后关系，且 v_i 只能在 v_j 之前。API 调用图 G，表示 API 之间的执行顺序以及目标程序的整体结构。

定义 2　（图同构）设 $G_1=(V_1,E_1)$ 和 $G_2=(V_2,E_2)$ 是 2 个简单图，若存在一个从 V_1 到 V_2 的双射函数 f，且 f 具有这样的性质，对 V_1 中的所有顶点 x 和 y，x 和 y 在 G_1 中相邻，当且仅当 $f(x)$和 $f(y)$ 在 G_2 中也相邻，那么称 G_1 和 G_2 是同构的（isomorphic），这样的函数 f 称为同构函数。

3.2　方法概述

本文提出的恶意代码同源性分析方法，整体框架如图 1 所示。首先，采用静态分析方法分析恶意代码，提取其 API 调用图，之后对 API 调用图进行处理，使其能够与卷积神经网络的输入匹配。

卷积神经网络处理图像时，作为输入，图像以像素为基本单位进行卷积操作。设 API 调用图 G，它包含 N 个基本节点 $G=\{g_1,g_2,\cdots,g_N\}$，本文主要任务是将图结构的数据进行转换，使其作为卷积神经网络输入。若以图结构数据作为卷积神经网络的输入，则会存在 2 个问题。

1) 不同恶意代码提取的 API 调用图，其规模不同，如果以卷积神经网络处理该数据，需要将图结构数据进行标准化，使输入数据的规模统一于卷积神经网络。

2) 由于图像具有隐性的空间顺序，卷积神经网络处理图像特征时，其感知野会以固定的方向和步长移动。而针对图结构数据，以节点作为处理的基本单位，需要规定移动的起始点以及卷积的移动规则。

针对数据规模不统一的问题，本文采用关键节点选择来统一数据规模使图数据结构标准化。同时，针对卷积神经网络感知野的构建，提出了一个关键节点邻域选择算法。以上 2 个问题将在第 4 节详细讨论。

4　关键节点选择及感知野构建

4.1　API 调用图提取

API 调用图即恶意代码在执行过程中，采用图结构表示 API 调用执行的先后关系，该方式能够更加全面地保留恶意代码各 API 执行的情况以及以 API 表示的整体特征信息。将恶意代码二进制文件转换成 API 调用图，采用图结构表示恶意代码的特征信息，其过程如图 2 所示。

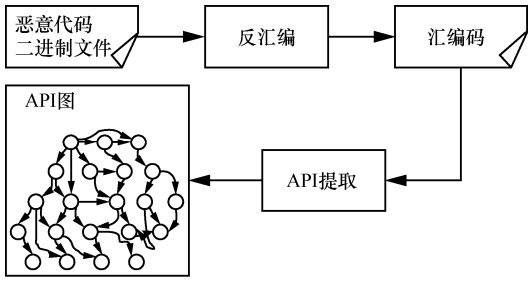


图 2　API 图提取过程

恶意代码的二进制文件经过反汇编得到其对应的汇编代码，将汇编代码按照基本块进行划分，分别扫描每个基本块，选取包含 call 指令的语句以及包含跳转指令的语句，如 jz、jmp、jnz 等。call 指令调用的函数分为两类，自定义函数和 API。如果调用目标为自定义函数，则进入该自定义函数内部，继续扫描其内部汇编语句，筛选其内部 API。筛选结束后，根据 API 执行的先后顺序和跳转指令的跳转结构，将不同函数以及不同基本块中 API 连

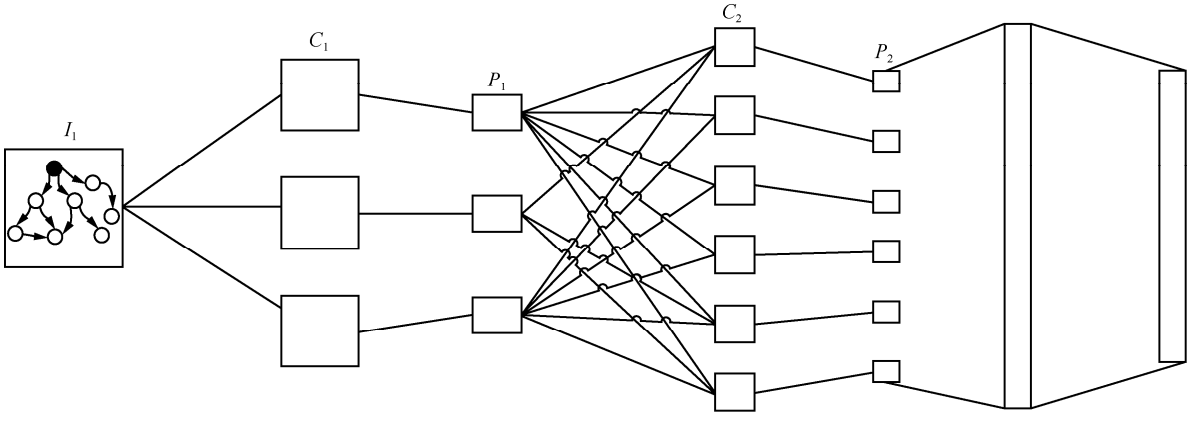


图 1　基于 CNN 的恶意代码同源性分析模型

接，建立恶意代码的整体 API 调用图。

4.2　关键节点选择

不同恶意代码的 API 调用图，规模会有一定差异，为了适应卷积神经网络的输入，需要对其规模进行标准化处理，确定输入节点的个数以及对应节点的感知野。文献[6]通过对图中节点的重要性进行排序，按照节点重要性的先后次序，确定关键节点。通常，图数据中关键节点的选择主要根据节点的重要程度，度量节点重要程度的方法包括节点度、接近度等。节点度是指图中与此节点连接边的数量。接近度则是对节点是否在图中心的一个度量，通过计算该节点与图中所有其他节点的距离来反映节点的中心程度。

节点度和接近度都是以节点外在属性来表示节点的重要程度，但本文针对 API 调用图选择重要节点，每个节点均表示不同的 API，具有不同的含义，所以不能只简单地根据节点的度数以及节点在图中的位置计算节点的重要程度。如果一个节点的重要程度比较高，但与它相连的节点并不重要，那么这个节点并不一定重要。相反，若一个节点重要程度不高，但与它相连的节点较为重要，那么该节点也可能具有较高的重要程度。

针对 API 调用图中节点重要性的计算，除了计算单个节点的重要程度，还要计算节点之间的相互关系。本文借鉴 PageRank 算法[20]，计算 API 调用图中节点的重要程度，同时对 API 进行分类，为不同 API 设置不同的等级，然后计算各节点在图中的重要程度，从而选择关键节点。

PageRank 算法的基本思想是，互联网中互相链接的网页，其重要程度能够相互反映。例如，当页面 A 存在指向页面 B 的链接，就认为 B 获得了 A 对它的贡献分值，该分值的多少取决于 A 本身的重要程度，即页面 A 的重要性越大，网页 B 贡献值计算过程中所得到贡献就越多。由于网页是相互指向，PageRank 的分值计算是一个迭代的过程，最终根据分值对网页进行检索排序。PageRank 分值计算式如下

$$PR(p_i)=\frac{1-q}{N}+q\sum_{p_j}\frac{PR(p_j)}{L(p_j)} \tag{1}$$

其中，$PR(p_i)$为页面 p_i 的 PageRank 值，N 为页面总数量，q 为用户随机到达一个页面的概率，$L(p_j)$为页面 p_j 链出页面的数量。通常初始页面的 *PR* 值为 1，根据式(1)递归计算各网页的 *PR* 值，直到各值趋于稳定。

计算 API 调用图中各节点的重要程度，可以利用 PageRank 算法，根据 API 调用图中各 API 的调用关系进行计算，但该方式与互联网计算网页重要性有一些差异。API 调用图中某个节点并不需要用户随机访问，因此式(1)中的随机概率 q 并不适用。另外，每个 API 对恶意代码行为的贡献程度有差异，所以在针对 API 调用图中的节点计算的过程中，需要将 API 本身对行为的贡献度考虑进去。

为计算 API 对恶意代码行为的贡献，需要考虑 API 是否单独出现在恶意代码中或正常程序中。借助代码在线分析平台 virustotal[21]，该平台采用沙箱机制，整合超过 90 种杀毒软件，对用户上传的样本进行分析，能够给出恶意性的评价。通过随机选取最近两年上传的样本 581 165 个，其中，恶意样本 340 023 个，正常样本 241 142 个，统计 API 在样本中出现的频率，并计算每个 API 的 TF-IDF[22]（term frequency-inverse document frequency）值，其计算式如下

$$tf\text{-}idf_{i,j}=\frac{n_{i,j}}{\sum_k n_{k,j}}\cdot\lg\frac{|D|}{|\{j:t_i\in d_j\}|+1} \tag{2}$$

其中，$n_{i,j}$ 为包含 API 调用 i 的样本在类别 j 中的总数，分母则为类别 j 的总样本个数，本文把样本分为两类，分别为正常样本和恶意样本。$|D|$为总样本数目，$|\{j:t_i\in d_j\}|$为包含 API 调用 i 的样本数目。

所以，在计算节点重要程度的过程中引入 TF-IDF，其计算式如下

$$R(v_i)=\frac{tf\text{-}idf_i}{N}+tf\text{-}idf_i\cdot\sum_{p_j}\frac{R(v_j)}{L(v_j)} \tag{3}$$

其中，$R(v_i)$为节点 v_i 的重要程度，由于只考虑 API 对恶意样本行为的贡献程度，所以 *TF-IDF* 值选择恶意样本类别的 *TF-IDF* 值，$L(p_j)$是节点 v_j 调用节点的数量。根据式(3)递归计算，当结果趋于稳定，则得到节点的重要程度。

4.3　感知野构建

已经确定关键节点，为了进行卷积操作，需要为每个关键节点确定一个感知野。以关键节点的邻域所包含的节点集合作为感知野的候选集。设感知野的大小为 k，关键节点集合 $R=\{r_1,r_2,\cdots,r_N\}$，通过设置规则，对邻域内的节点进行筛选。

设置一个统一的规则，使图的邻域到感知野是一条单射函数，从而保证具有相似结构特征和属性的节点能够映射到感知野的相似区域，这个过程是对关键节点及其邻近节点的一个规范化的过程。

本文以 4.2 节得到的 581 165 个样本所包含 API 的 *TF-IDF* 值作为邻域节点选择优先级的判断条件，采用层次遍历的方式，对关键节点的邻域进行遍历，并根据感知野大小，选择候选节点，建立感知野，感知野建立算法如算法 1 所示。

算法 1 感知野建立算法

输入 API 调用图 G，关键节点集合 $R=\{r_1, r_2,\cdots,r_N\}$，感知野大小 k

输出 API 调用图的感知野集合 P

1) for($i=1;i<N;i++$)
2) $B=BFS(G,r_i,1)$;
3) $B'=Sort\_TFIDF(B)$;
4) if $len(B')\geqslant k$
5) $p_j=B'[0:k]$;
6) else
7) $k'=k-len(B')$;
8) for ($j=0;j<len(B');j++$)
9) $B_2=BFS(G,b_j,1)$;
10) $B_2'=Sort\_TFIDF(B_2)$;
11) if $len(B_2')\geqslant k$
12) $p_j=p_j+B'[0:k]$
13) break;
14) end if
15) end for
16) end if
17) end for

感知野建立算法 1)~5)行说明如下，首先扫描关键节点，遍历关键节点集合，对每个关键节点进行层次遍历得到节点集合 B。然后对 B 中节点按照 *TF-IDF* 值进行排序得到集合 B'，如果 B' 的长度大于感知野大小 k，则以 B' 的前 k 个节点构建感知野。若 B' 的长度小于 k，则以 B' 中的节点进行层次遍历，继续按照上述流程向感知野中加入节点，算法 8)~13)行，表示判断和再遍历的过程。

5 实验及分析

5.1 实验准备

本文实验数据集主要通过 VxHeaven 网站收集得到，该数据集包含 27 万个经过标记的样本，本文主要关注 Windows 平台下 32 位可执行文件。由于数据集中各样本家族数目不相同，选取其中样本数最多的 8 个家族，如表 1 所示。每个家族选择 120 个样本，其中，100 个样本构成训练集，20 个样本构成测试集。针对每个样本，利用 IDA 反汇编得到恶意代码的汇编指令，并根据汇编指令构建 API 序列图。

表 1 样本家族

编号	家族名称
1	Backdoor.Win32.Hupigon
2	Backdoor.Win32.Agent
3	Backdoor.Win32.PcClient
4	Trojan.Win32.Agent
5	Backdoor.Win32.Bifrose
6	Backdoor.Win32.Poison
7	Rootkit.Win32.Agent
8	Hoax.Win32.Renos

5.2 实验设计

本文主要从 2 个方面进行实验：不同关键节点选择算法对同源性判定结果的影响；本文方法检测准确率以及与不同恶意代码检测算法进行比较。首先，计算采用不同关键节点选择算法进行实验，对比不同节点选择算法对同源性分析检测结果的影响。然后，将本文算法在不同数据训练集规模下训练得到的模型，用于检测恶意代码，并对比本文算法与现有方法检测效果的差异。每种实验方法，均进行 10 次实验，以保证实验的随机性。

在评价检测效果方面，使用误报率（false positive rate）和检测率（true positive rate）来评价检测结果，使用准确率（accuracy）和错误率（error）对比不同方法的检测效果。各检测效果主要针对不同家族样本被识别为本家族的情况。具体来说，假设在被检测方法判断为家族 a 中，实际不属于家族 a 的数量为 *FP*，实际属于家族的数量为 *TP*；在被检测方法判断不属于家族 a 的样本中，实际为不属于家族 a 的数量为 *TN*，实际为家族 a 的数量为 *FN*。误报率、检测率、准确率、错误率的定义如下

$$\text{误报率}=\frac{FP}{TN+FP} \tag{4}$$

$$检测率=\frac{TP}{TP+FN} \tag{5}$$

$$准确率=\frac{TP+TN}{TP+FN+TN+FP} \tag{6}$$

$$错误率=\frac{FP+FN}{TP+FN+TN+FP} \tag{7}$$

5.3　实验结果

5.3.1　关键节点选择对结果影响

通过实验对比 4.2 节提出的关键节点选择算法所选择的关键节点对同源性判定的影响，分别使用了 PageRank 算法、节点接近度、随机选择节点以及本文所提出的结合 PageRank 和 TF-IDF 的节点选择算法（Page&TF）。同时，也比较了不同关键节点数量对检测结果的影响，也就是算法 1 所表示的关键节点集合的大小 N。图 3 和图 4 分别为初始训练集每个类别分别包含 50 个样本以及 100 个样本的情况下，10 次实验平均准确率。

图 3 和图 4 的纵坐标均表示准确率，横坐标表示节点规模。从图 3 和图 4 中可以看出，随着节点规模的不断增加，除了随机选择方法，其他 3 种节点选择方法的准确率不断提高，并且本文所采用 PageRank 和 TF-IDF 相结合的算法节点选择的有效性最好，在不同节点规模的情况下，其检测准确率均超过 0.8。随着训练集样本数的提升，在样本充足的情况下，本文提出方法的准确率在节点数为 144 时达到了 0.9，说明本文方法在准确率方面也有较好效果，并且随着节点规模超过 144，各节点选择算法的准确率均没有太大提升。

图 5 和图 6 显示了关键节点数为 144 的情况下，4 种节点选择算法随着初始训练样本数提升得到的检测率和误报率的情况。

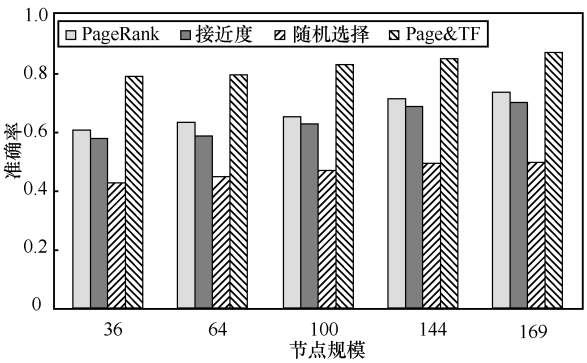


图 3　不同关键节点选择算法对准确率的影响（样本数 50）

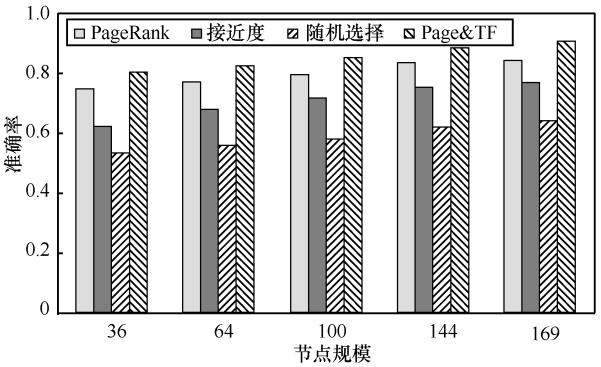


图 4　不同关键节点选择算法对准确率的影响（样本数 100）

图 5 的纵坐标表示检测率，横坐标表示训练集的规模。从图 5 可以看出，3 种关键节点算法随着训练集规模提升，其检测率均有明显提升，而本文所使用方法在检测率方面均超过 0.8，并且随着规模的提升，最终检测率达到 0.9。图 6 纵坐标表示误报率，其反映不同节点选择算法的误报率的情况，各算法误报率随着训练样本的提升，均有不同程度下降，本文所使用方法与 PageRank 算法均有较好的误报率。综上分析，本文结合了 PageRank 和 TF-IDF 算法的关键节点选择算法，对同源性分析具有较好的促进所用，使同源性分析的准确率以及检测率均具有较好的效果。

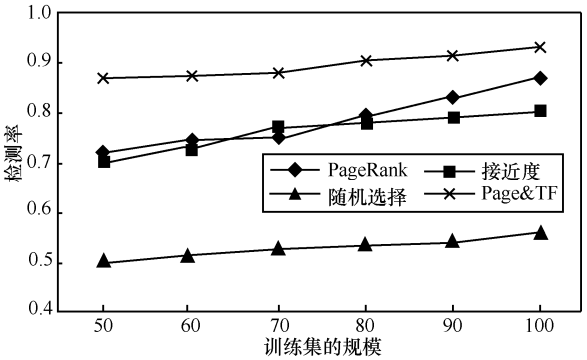


图 5　不同关键节点选择算法的检测率

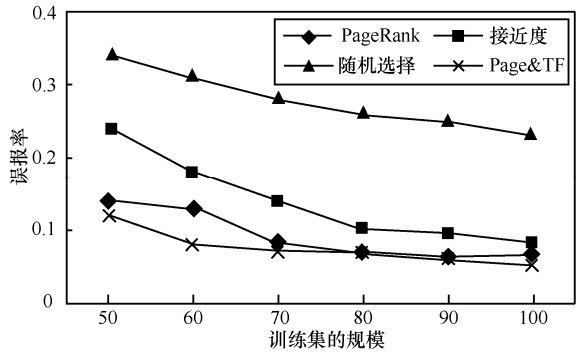


图 6　不同关键节点选择算法的误报率

5.3.2 检测结果对比

恶意代码检测率是识别恶意代码检测方法好坏的关键，本文将表 1 提供的 8 个家族的恶意代码合并，总共 800 个恶意样本作为训练集。同时，从这 8 个家族中每个家族另外选择 20 个样本共 200 个样本作为测试集中恶意代码，与随机收集到的 Windows 平台下 32 位正常软件的 200 个样本组成测试集进行实验。实验结果如表 2 所示。

表 2 不同训练集规模下的检测准确率

训练集规模/个	准确率	误报率
200	0.91	0.07
400	0.923	0.062
600	0.942	0.05
800	0.967	0.041

表 2 中分别从训练样本中随机抽取一定数量样本作为训练集，对卷积神经网络进行训练，最终对恶意代码检测的准确率均超过 0.9。并且在训练集规模为 800 时，准确率达到 0.96。另外，将本文所提方法与现有恶意代码检测方法进行对比，其结果如表 3 所示。

表 3 方法对比

使用方法	准确率
Byte Code n-gram + Opcode n-gram	0.96
Opcode n-gram + API	0.962
API graph + CNN	0.967

表 3 分别列出了不同恶意代码检测方法所采用的技术，分别是操作码序列和字符串信息的 n-gram 方法[23]，操作码序列和 API 方法结合的 n-gram 方法[24]，以及本文所使用的 API 调用图和卷积神经网络相结合的方法。通过对各种方法准确率进行分析，本文方法的准确率达到 96.7%，说明利用卷积神经网络在恶意代码检测方面具有较好的效果。

6 结束语

以 API 调用图为代表的图结构特征除了能够有效表示恶意代码行为信息外，还具有较好的整体特性。但由于基于图的匹配算法复杂度较高，不利于恶意代码的实时检测。本文利用卷积神经网络，以 API 调用图作为分析目标，通过设计关键节点选择算法，提取关键节点，并以关键节点的邻域，设计了邻域节点选择算法，构建关键节点的感知野，以此作为卷积神经网络的输入。通过对 Windows 平台 32 位的 800 个恶意代码样本组成的训练集进行训练，并对测试集进行测试，恶意代码检测的准确率达到 96%，同时同源性判定的准确率达到 92%。这表明该方法在恶意代码检测和同源性分析方面均有较好的效果。

卷积神经网络的发展迅速，本文只利用了简单的卷积神经网络，如果通过实验，找到更加适合恶意代码检测的卷积神经网络结构，将对进一步提升恶意代码检测和同源性分析的效果具有重要价值。

参考文献：

[1] SAXE J, BERLIN K. Deep neural network based malware detection using two dimensional binary program features[C]//International Conference on Malicious and Unwanted Software. 2015:11-20.

[2] SANTOS I, BREZO F, UGARTE-PEDRERO X, et al. Opcode sequences as representation of executables for data-mining-based unknown malware detection[J]. Information Sciences, 2013, 231(9): 64-82.

[3] GUPTA S, SHARMA H, KAUR S. Malware characterization using windows API call sequences[C]//International Conference on Security, Privacy, and Applied Cryptography Engineering. Springer International Publishing, 2016:271-280.

[4] NARAYANAN A, MENG G, YANG L, et al. Contextual Weisfeiler-Lehman graph kernel for malware detection[C]// International Joint Conference on Neural Networks. 2016:4701-4708.

[5] EPPSTEIN D. Subgraph isomorphism in planar graphs and related problems[J]. SODA'95 Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms, 1999, 3(3):332-346.

[6] NIEPERT M, AHMED M, KUTZKOV K. Learning convolutional neural networks for graphs[C]//International Conference on Machine Learning, 2016: 2014-2023.

[7] DUVENAUD D K, MACLAURIN D, IPARRAGUIRRE J, et al. Convolutional networks on graphs for learning molecular fingerprints[C]//Advances in Neural Information Processing Systems. 2015: 2224-2232.

[8] KIPF T N, WELLING M. Semi-supervised classification with graph convolutional networks[J]. arXiv preprint arXiv:1609.02907, 2016.

[9] DEFFERRARD M, BRESSON X, VANDERGHEYNST P. Convolutional neural networks on graphs with fast localized spectral filtering[C]//Advances in Neural Information Processing Systems, 2016: 3844-3852.

[10] KI Y, KIM E, KIM H K. A novel approach to detect malware based on API call sequence analysis[J]. International Journal of Distributed Sensor Networks, 2015, 11(6): 659101.

[11] THOMPSON J D, GIBSON T J, HIGGINS D G. Multiple sequence alignment using CrustalW and clustalX[J]. Current protocols in bioinformatics, 2002.

[12] LEE T, CHOI B, SHIN Y, et al. Automatic malware mutant detection and group classification based on the n-gram and clustering coefficient[J]. The Journal of Supercomputing, 2015: 1-15.

[13] OKTAVIANTO D, MUHARDIANTO I. Cuckoo malware analysis[M]. Packt Publishing, 2013.

[14] CESARE S, XIANG Y, ZHOU W. Malwise-an effective and efficient classification system for packed and polymorphic malware[J]. IEEE Transactions on Computers, 2013, 62(6):1193-1206.

[15] PARK Y, REEVES D, MULUKUTLA V, et al. Fast malware classification by automated behavioral graph matching[C]// AMIA Annu Symp Proc, 2010:1-4.

[16] KINABLE J, KOSTAKIS O. Malware classification based on call graph clustering[J]. Journal of Computer Virology and Hacking Techniques, 2011, 7(4):233-245.

[17] HASSEN M, CHAN P K. scalable function call graph-based malware classification[C]//The Seventh ACM on Conference on Data and Application Security and Privacy. 2017: 239-248.

[18] 杨帆, 张焕国, 傅建明,等. 基于图编辑距离的恶意代码检测[J]. 武汉: 武汉大学学报(理学版), 2013, 59(5):453-457.
YANG F, ZHANG F G, FU J M, et al. Malware Detection Based on Graph Edit Distance[J]. Wuhan: Wuhan Univ (Nat Sci Ed.) , 2013, 59(5): 453-457.

[19] 刘星, 唐勇. 恶意代码的函数调用图相似性分析[J]. 计算机工程与科学, 2014, 36(3):481-486.
LIU X, TANG Y. Similarity analysis of malware's function-call graphs[J]. Computer Engineering & Science, 2014, 36(3):481-486.

[20] ARASU A. Hector garcia-molina, andreas paepcke, and sriram raghavan. searching the Web[J]. ACM Transactions on Internet Technology, 2001, 1:2-43.

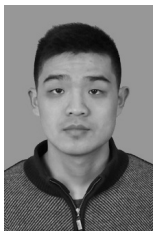
[21] TOTAL V. VirusTotal-Free online virus, malware and URL scanner[J]. 2012.

[22] WU H C, LUK R W P, WONG K F, et al. Interpreting TF-IDF term weights as making relevance decisions[J]. ACM Transactions on Information Systems, 2008, 26(3):55-59.

[23] SANTOS I, BREZO F, UGARTE-PEDRERO X, et al. Opcode sequences as representation of executables for data-mining-based unknown malware detection[J]. Information Sciences, 2013, 231(9): 64-82.

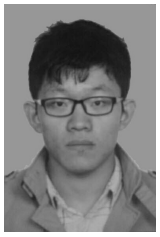
[24] SANTOS I, DEVESA J, BREZO F, et al. OPEM: a static-dynamic approach for machine-learning-based malware detection[M]// International Joint Conference CISIS'12-ICEUTE'12-SOCO'12 Special Sessions. Springer Berlin Heidelberg, 2013:271-280.

作者简介：



赵炳麟（1990-），男，河南洛阳人，数学工程与先进计算国家重点实验室博士生，主要研究方向为信息安全、代码安全性分析。

孟曦（1992-），女，山东济宁人，数学工程与先进计算国家重点实验室硕士生，主要研究方向为信息安全、恶意代码分析。



韩金（1993-），男，安徽蚌埠人，数学工程与先进计算国家重点实验室硕士生，主要研究方向为信息安全、恶意代码分析。

王婧（1985-），男，陕西杨凌人，数学工程与先进计算国家重点实验室讲师，主要研究方向为计算机应用与技术。

刘福东（1986-），男，辽宁沈阳人，博士，数学工程与先进计算国家重点实验室讲师，主要研究方向为并行与分布式系统。