Deep android malware detection(安卓恶意软件的深度学习检测)

> McLaughlin N, Martinez del Rincon J, Kang B J, et al. Deep android malware detection[C]//Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy. 2017: 301-308.

> Keywords： Malware Detection, Android, Deep Learning

Summary

- 这篇论文使用二进制程序反编译的静态操作码作为该程序的特征，使用 `CNN (convolutional neural network)` 为模型，取得了较好的效果，使用 `accuracy` 、 `precision` 、 `recall` 、 `f-score` 等评价标准来评价该模型。在本文的数据集实验环境下，F-score=0.97，在实际的数据集环境中，F-score=0.71。 虽然实际环境中的效果不是特别好，但是模型训练所需要的时间相对较少，效率较高，而且内存使用率是一个常数。
- 它不需要进行动态特征提取，框架更加简单
- 比n-gram相比，效率提高了很多，虽然真正的效果一般，但是具有创新性，对论文的撰写来说，提供了一种「尝试将已有的处理其他方面的机器学习算法用于我们想要实现的目标上」的思路，说不定会获得不错的效果

Glossary

- android malware:基于安卓平台的恶意软件
- CNN：由3个部分构成
  - 卷积层：提取局部特征(filters和激活函数构成)
  - 池化层：大幅降低参数量级(降维)
  - 全连接层：类似于传统神经网络的部分，用来输出想要的结果
- opcode：操作码，机器指令中的指令部分

Research Objective(s)

- apply convolutional neural networks to the problem of malware detection.（将CNN应用于安卓恶意软件的检测）

Background / Problem Statement

- many of these methods are reliant on expert analysis to design the discriminative features that are passed to the machine learning system used to make the final classification decision. --当前大部分的恶意软件检测的方法依赖于特征的选择和设计
- Recently, convolutional networks have been shown to perform well on a variety of tasks related to natural language processing -- CNN在自然语言处理方面表现良好
- 考虑选择静态特征-操作码，并结合CNN进行训练

Method(s)

Disassembly of Android Application

- 特征提取

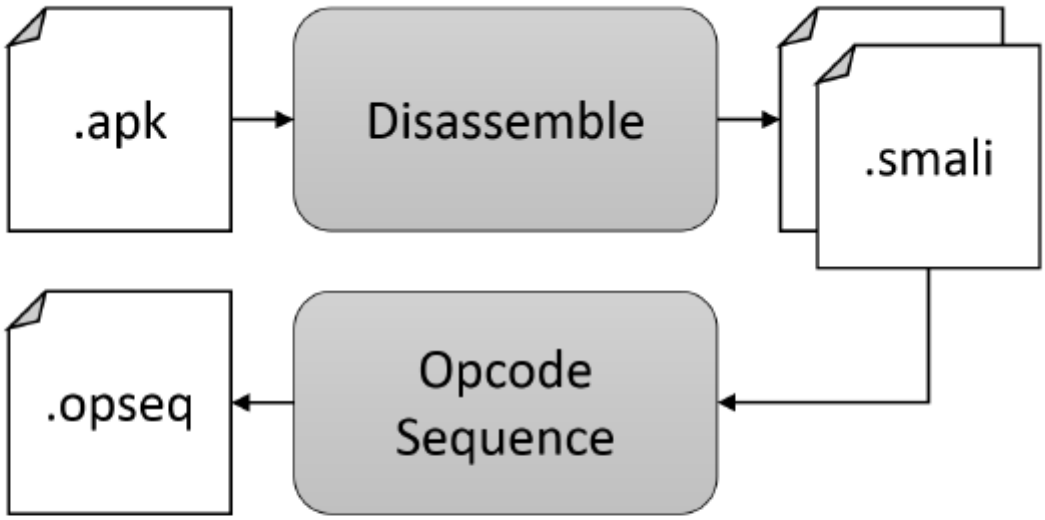

Figure 1: Work-flow of how an Android application is disassembled to produce an opcode sequence.

> 先将apk文件反编译，得到apk的所有资源文件(其中的.smali文件内的语法使用Dalvik虚拟机指令语言)。 之后，从.smali文件中提取每个method的操作码序列，组合得到当前smali文件(一个class)的特征数据集。 最后，所有的smali文件提取的操作码序列汇集得到单个apk文件的操作码序列。

Network Architecture

- 整体框架图

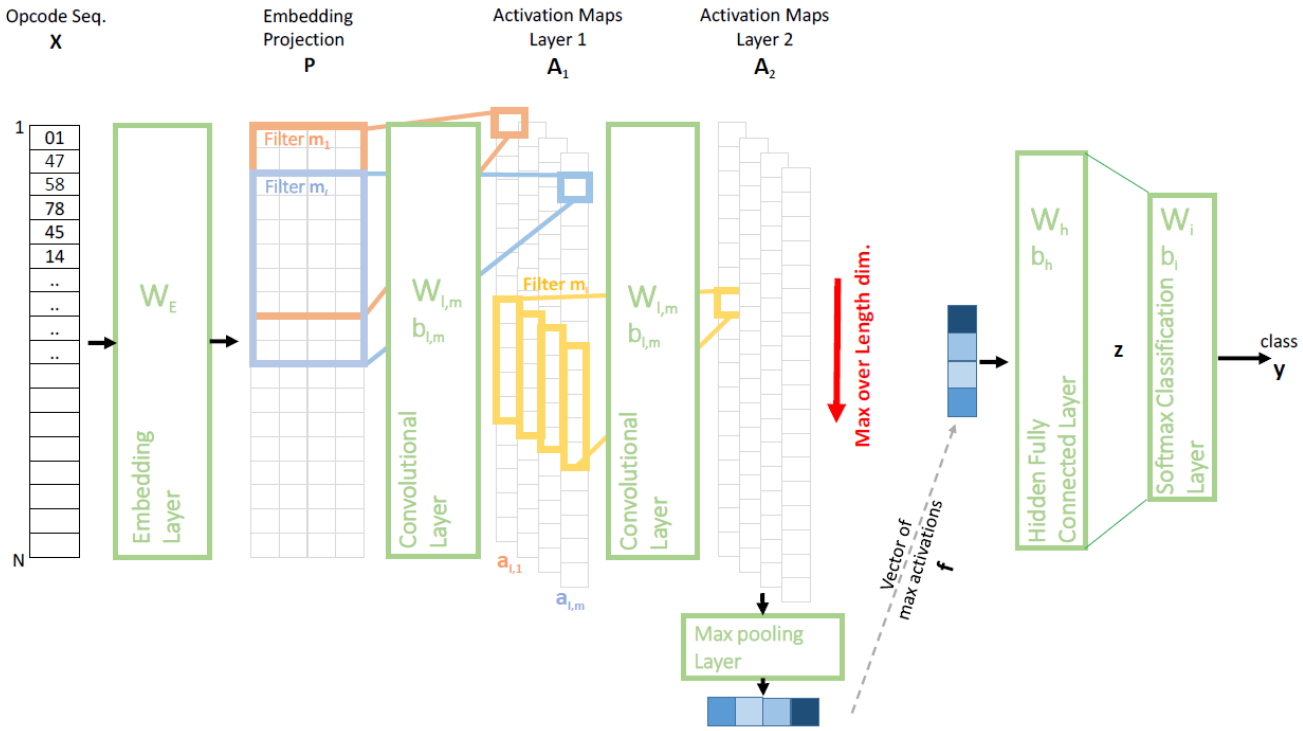

Figure 2: Malware Detection Network Architecture.

Opcode Embedding Layer

- one-hot编码操作码序列，X[n]是第n个向量表示的操作码；通过查看Dalvik文档，向量的维数(dimension)是218维。 X[n]乘一个权重矩阵W(随机初始化)后，将操作码映射到一个k维的Embedding空间中，得到P[i]矩阵。

$$p_i = x_i W_E$$

> 这样做的目的是：让网络可以在一个连续的k维空间中学习到每个操作码的正确表示。Embedding空间的维数会影响网络表示映射关系的能力，因此，使用更高的维度可以使网络具有更强的灵活性去学习高维的非线性映射关系。

Convolutional Layers

- 第一个卷积层接受 `N*K` 的Embedding矩阵作为输入，更深层次的卷积层以之前卷积层的输出作为输入。 每个卷积层有M[l]个filter(第一层size是 `s[1]*k` ，更深层的size是 `s[l]*M[l-1]`)，这意味着第一层的filter可以检测s[1]个操作码。 在样例前向传播的时候，每个filter都会产生一个激活映射Al,m，所有的filter产生的组合在一起，构成矩阵Al 第一个卷积层的filter产生的矩阵P可以用下面的第一个公式表示:

$$a_{l,m} = relu(Conv(P)_{W_{l,m},b_{l,m}}) \qquad (2)$$

- 在更深层次的卷积层中，使用A[l-1]作为l层的输入

$$A_l = [a_{l,1} \,|\, a_{l,2} \,|\, ... \,|\, a_{l,m}] \qquad (3)$$

- 经过最后的卷积层后，接着使用下面的公式进行maxpooling，使得f的长度是M[l]（本文中的M[l]值为2）。

$$f = [\max(a_{L,1}) \,|\, \max(a_{L,2}) \,|\, ... \,|\, \max(a_{L,m})] \qquad (4)$$

Classification Layers

- 经过神经网络训练以后输出 z

$$z = relu(W_h f + b_h) \qquad (5)$$

- 使用 `softmax` （归一化函数）得出分类

$$p(y = i|z) = \frac{\exp(w_i^T z + b_i)}{\sum_{i'=1}^{I} \exp(w_{i'}^T z + b_{i'})} \qquad (6)$$

- 卷积层的结果向量f，作为多层感知机(MLP)的输入-一个隐藏层和一个输出层。 其中隐藏层使用Relu作为激活函数，最后输出Softmax归一化的概率值。

Learning process

- 训练的损失函数如下

$$C = -\frac{1}{b} \sum_{j=1}^{b} \sum_{i=1}^{I} 1\{y^{(j)} = i\} log\, p(y^{(j)} = i|z^{(j)}) \qquad (7)$$

- 梯度下降方式如下，同时梯度的权重和样本的数量成反比，防止出现由于某类样本的数量过多导致检测结果偏向这类的结果,其中a是学习率，为了使损失函数达到最小值，达到最好的训练效果

$$\Theta^{(t+1)} = \Theta^{(t)} - \alpha \frac{\partial C}{\partial \Theta} \qquad (8)$$

- 权重如下

```
if num(malware)=M,num(benign)=B,M<B
weight(malware)=1-M/(M+B)
weight(benign)=M/(M+B)
```

Evaluation

datasets

- Small Dataset: Android Malware Genome project - 863 benign & 1260 malware
- Large Dataset: McAfee Labs - 3627 benign & 2475 malware
- V. Large Dataset: Mcfee Labs - 9268 benign & 9902 malware

results

- full results in datasets aforesaid

Classification System	Feature Types	Benign	Malware	Acc.	Prec.	Recall	F-score
Ours (Small DS)	CNN applied to raw opcodes	863	1260	0.98	0.99	0.95	0.97
Ours (Large DS)	CNN applied to raw opcodes	3627	2475	0.80	0.72	0.85	0.78
Ours (V. Large DS)	CNN applied to raw opcodes	9268	9902	0.87	0.87	0.85	0.86
n-grams (Small DS)	opcode n-grams (n=1)	863	1260	0.95	0.95	0.95	0.95
	opcode n-grams (n=2)	863	1260	0.98	0.98	0.98	0.98
	opcode n-grams (n=3)	863	1260	0.98	0.98	0.98	0.98
n-grams (Large DS)	opcode n-grams (n=1)	3627	2475	0.80	0.81	0.80	0.80
	opcode n-grams (n=2)	3627	2475	0.81	0.83	0.82	0.82
	opcode n-grams (n=3)	3627	2475	0.82	0.83	0.82	0.82
DroidDetective [13]	Perms. combination	741	1260	0.96	0.89	0.96	0.92
Yerima [23]	API calls, Perms., intents, cmnds	1000	1000	0.91	0.94	0.91	0.92
Jerome [10]	opcode n-grams	1260	1246	-	-	-	0.98
Yerima [25] *	API calls, Perms., intents, cmnds	2925	3938	0.97	0.98	0.97	0.97
Yerima (2) [24]*	API calls, Perms., intents, cmnds.	2925	3938	0.96	0.96	0.96	0.96

Table 1: Malware classification results for our system on both the small and large datasets compared with results from the literature. Results from the literature marked with a (*) use malware from the McAfee Labs dataset i.e. our large dataset, while all others use malware sampled from the Android Malware Genome project [28] dataset i.e. our small dataset

- Realistic Testing results

Classification System	Acc.	Prec.	Recall	F-score
Ours	0.69	0.67	0.74	0.71

Table 3: Malware classification results of our system tested on an independent dataset of benign and malware Android applications.

- Realistic Testing results

System	Time per program (s)	Programs per second
Ours	0.000329	3039.8
1-gram	0.000569	1758.3
2-gram	0.010711	93.4
3-gram	0.172749	5.8

Table 2: Comparing the time taken to reach a classification decision and number of programs that can be classified per second, for our proposed neural network system and a conventional n-gram based system.

- Learning Curves

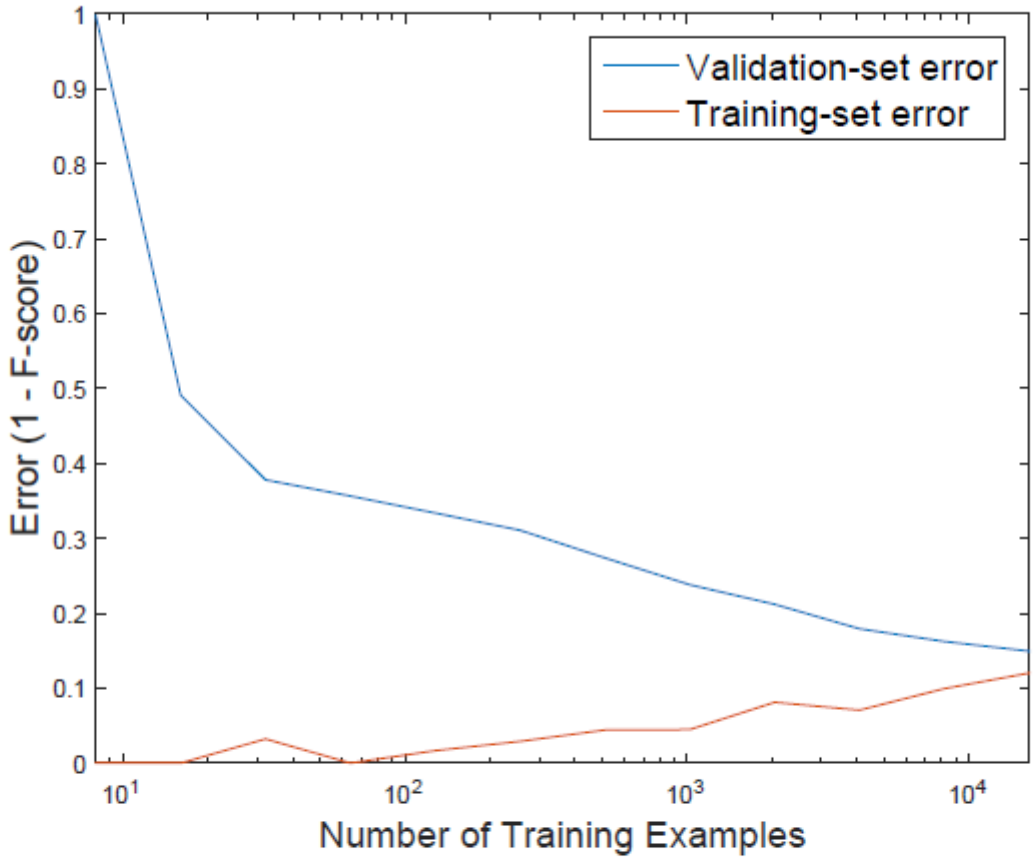

Figure 3: Learning curves for the Validation-set and Training-set as the number of training examples is varied. Note the log-scale on the x-axis.

> 学习曲线，1-F-score 和训练样本的关系

Conclusion

- 提出了一种新颖的 Android 恶意软件基于深度神经网络的检测系统，并已在四个不同的 Android 恶意软件数据集中进行了验证，发现有良好的效果。
- 能够同时训练大量特征，且仅需给出大量标记恶意软件样本的操作码序列。
- 不再需要人工设计特征提取，且比现有的基于 n-gram 的计算效率更高，并且可以实现在GPU上运行。