

Android Malware Detection using Ada-Boost Neural Networks

Monika Patidar¹; Ravi Khatri²

M.Tech Scholar (Cyber Security)¹, Professor and Head (Computer Science)², VITM Indore^{1,2}

*monikapatidar06@gmail.com*¹; *k.ravi@vitmindore.com*²

ABSTRACT

With the ever increasing popularity of Android powered smart phones, the security threats towards it have also escalated. In order to prevent such malwares, it is important to have accurate and deep understanding of them so that security measures to protect users' data could be taken accordingly. There are large numbers of attack scenarios where an attacker can compromise a user's data by taking advantage of the vulnerabilities of Android operating system. The proposed work uses an ada boost neural architecture of the SCG and KNN based neural networks and attains a classification accuracy of 97.5% which is significantly higher than previously existing techniques.

Keywords: Android, Malware, Artificial Neural Networks, Ada Boost, Scaled Conjugate Gradient, K Nearest Neighbor (KNN).

1. INTRODUCTION

Android platform offers sophisticated functionalities at very low cost and has become the most popular operating system for handheld devices. Apart from the Android popularity, it has become the main target for attackers and malware developers. The official Android market hosts millions of applications that are being downloaded by the users in a large number everyday. Android offers an open market model where no any application is verified by any security expert and this makes Android an easy target for developers to embed malicious content into their applications. The users sensitive data can be easily compromised and can be transferred to other servers. Furthermore, the existence of third party application stores contributes in spreading malwares for Android because Google Play also hosts the applications of third-party developers. Malware, as a malicious application that can be installed on mobile devices, can gain access to

these devices and collect user sensitive information. Malware has proven to be a serious problem for the Android platform because malicious applications can be distributed to mobile devices through an application market.

2. RELATED WORK

In 2018 IEEE, Hossein Sayadi et al. [1] present Ensemble Learning for Effective Run-Time Hardware-Based Malware Detection: A Comprehensive Analysis and Classification. Malware detection at the hardware level has emerged recently as a promising solution to improve the security of computing systems. Hardware-based malware detectors take advantage of Machine Learning (ML) classifiers to detect pattern of malicious applications at run-time. These ML classifiers are trained using low-level features such as processor Hardware Performance Counters (HPCs) data which are captured at run-time to appropriately represent the application behaviour.

In 2018 ELSEVIER, El Mouatez Billah Karbab et al. [2] present MalDozer: Automatic framework for android malware detection using deep learning. Android OS experiences a blazing popularity since the last few years. This predominant platform has established itself not only in the mobile world but also in the Internet of Things (IoT) devices. This popularity, however, comes at the expense of security, as it has become a tempting target of malicious apps. Hence, there is an increasing need for sophisticated, automatic, and portable malware detection solutions. This paper proposes MalDozer, an automatic Android malware detection and family attribution framework that relies on sequences classification using deep learning techniques.

In 2017 ACM, Shifu Hou et al. [3] propose Deep Neural Networks for Automatic Android Malware Detection. Because of the explosive growth of Android malware and due to the severity of its damages, the

detection of Android malware has become an increasing important topic in cybersecurity. Currently, the major defense against Android malware is commercial mobile security products which mainly use signature-based method for detection. However, attackers can easily devise methods, such as obfuscation and repackaging, to evade the detection, which calls for new defensive techniques that are harder to evade. In this paper, resting on the analysis of Application Programming Interface (API) calls extracted from the smali files, we further categorize the API calls which belong to the some method in the smali code into a block.

In 2017 IEEE, R. Vinayakumar et al. [4] propose Deep android malware detection and classification. Long short-term memory recurrent neural network (LSTM-RNN) have witnessed as a powerful approach for capturing long-range temporal dependencies in sequences of arbitrary length. This paper seeks to model a large set of Android permissions particularly the permissions from Normal, Dangerous, Signature and Signature Or System categories within a large number of Android application package (APK) files of Cyber Security Data Mining Competition (CDMC 2016), Android malware classification challenge.

In 2016 IJC, Sergei Bezobrazov et al. [5] proposed The Methods of Artificial Intelligence for Malicious Applications Detection in Android Os. his paper presents and discusses a method for Android's applications classification with the purpose of malware detection. Based on the application of an Artificial Immune System and Artificial Neural Networks we propose the "antivirus" system especially for Android system that can detect and block undesirable and malicious applications. This system can be characterized by self-adaption and self-evolution and can detect even unknown and previously unseen malicious applications. The proposed system is the part of our team's big project named "Intelligent Cyber Defense System" that includes malware detection and classification module, intrusions detection and classification module, cloud security module and personal cryptography module.

3. Artificial Neural Networks

Artificial neural networks are a practical way of implementing artificial intelligence with an aim to solve fitting problems generally needing herculean efforts due to the data size and complexity.

The artificial neural architecture tries to imitate the human thought process in the following ways:

- Process data as a parallel stream independently
- Identifying patterns and correlating them.
- Evolving and updating the experiences (called weights) as per the changes in the data received. Neural networks work on training and testing mechanism.
- Finally rendering an output and further storing it as an experience.

ANN basically tries to inherit this capability of human brain to self-train itself for tasks which are never been performed by it that too very efficiently. Human brain's structure consists of neurons which are interconnected with each other and there by forming a very large network which is well connected thereby helps in performing very complex task like voice and image recognition very easily. The same task when performed using normal computer won't give accurate result. Hence ANN mimics neurons structure of human brain to discover link between input and targets. Neurons have this ability to save previous experimental data. The speed of human brain is several thousand times faster than traditional computer because in brain unlike traditional computer as whole information is not passed from neuron to neuron they are rather encoded in the neuron network. This is reason why neural network is also named as connectionism.

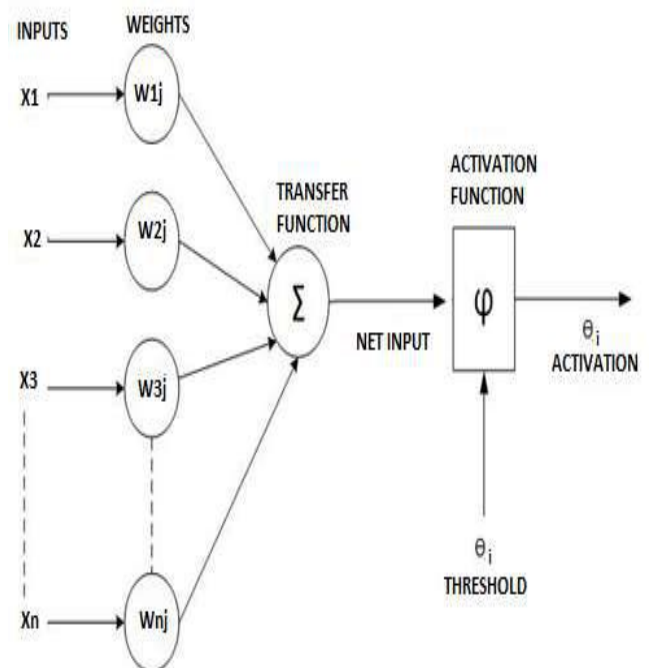


Fig 1: Mathematical Model of ANN

The mathematical conversion of the ANN can be done by analyzing the biological structure of ANN. In the above example, the enunciated properties of the ANN that have been emphasized upon are:

1. Strength to process information in parallel way.
2. The power to grasp and learn from weights
3. Searching for patterned sets in complex models of data.

Consider a signal S_1 travelling through a path p_1 from dendrites with weight w_1 to the neuron. Then the value of signal reaching the neuron will be $S_1 \cdot w_1$. If there are "n" such signals travelling through n different paths with weights ranging from w_1 to w_n and the neuron has an internal firing threshold value of θ_n , then the total activation function of the neuron is given by:

$$Y = \sum_{i=1}^n X_i \cdot W_i + \theta_i \quad (1)$$

X_i represents the signals arriving through various paths,

W_i represents the weight corresponding to the various paths and

θ is the bias.

The entire mathematical model of the neuron or the neural network can be visualized pictorially or the pictorial model can be mathematically modelled. The design of the neural network can be modeled mathematically and the more complex the neural design, more is the complexity of the tasks that can be accomplished by the neural network. The soul of the above model lies in the fact that the system so developed tries to mimic the working of human brain in terms of the following:

- It works in a complex parallel computation manner
- High speed of performance due to the parallel architecture.
- It learning and adapt according to the modified link weights.

4. PROPOSED METHODOLOGY

4.1 Scaled Conjugate Gradient Algorithm

The major advantages of the scaled conjugate gradient algorithm are:

- 1) Low space complexity

- 2) Relatively low time complexity suited to real time applications.

Conjugate Gradient (CG) algorithm is a modified version of steepest descent algorithm. In CG, a hunt is done in such a direction so as to generate a faster convergence than the steepest decent direction, while saving the error minimization attained in all previous steps. The direction in which CG moves to update weights and bias is called the conjugate direction. In linear CG algorithms, the step-size (total data processed per step) is updated in each iteration. At the start of the algorithm, the direction taken in to account is the direction of the steepest descent [20]. This is done only till the first iteration is complete.

$$p_0 = -g_0 \quad (2)$$

The weights are updated as follows

$$x_{k+1} = x_k + \alpha_k g_k \quad (3)$$

Where, α_k is the determined step size and

$$P_k = -g_k + \beta_k P_{k-1} \quad (4)$$

Here p is search direction vector and g is gradient direction vector.

There are various versions of Conjugate Gradient algorithms which can be categorized by the manner in which the factor β_k is calculated. In this study, we have used Scaled Conjugate Gradient (SCG) Algorithm, in which we use LM algorithm combined with CG algorithm to calculate step size, unlike only the line search technique in the CG approach.

For SCG, β_k factor calculation and direction of the new search can be shown as in following equations.

$$\beta_K = \frac{(|g_{k+1}|^2 - g_{k+1}^T g_k)}{g_k^T g_k} \quad (5)$$

$$P_{k+1} = -g_{k+1} + \beta_k P_k \quad (6)$$

Design parameters are updated at each iteration user independently, which is crucial for the success of the algorithm. This is a major advantage compared to the line search based algorithms.

4.2 The KNN Approach

KNN stands for the K-Nearest Neighbor. KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique we generally look at 3 important aspects:

1. Ease to interpret output
2. Calculation time
3. Predictive Power

The KNN works on the principle of finding the Euclidean distance of the present data sample to a particular class. The steps involved in KNN are:

- 1) Receive an unclassified data;
- 2) Measure the distance from the new data to all others data that is already classified;
- 3) Gets the K smaller distances;
- 4) Check the list of classes had the shortest distance and count the amount of each class that appears;
- 5) Takes as correct class the class that appeared the most times;
- 6) Classifies the new data with the class that was done in step 5.

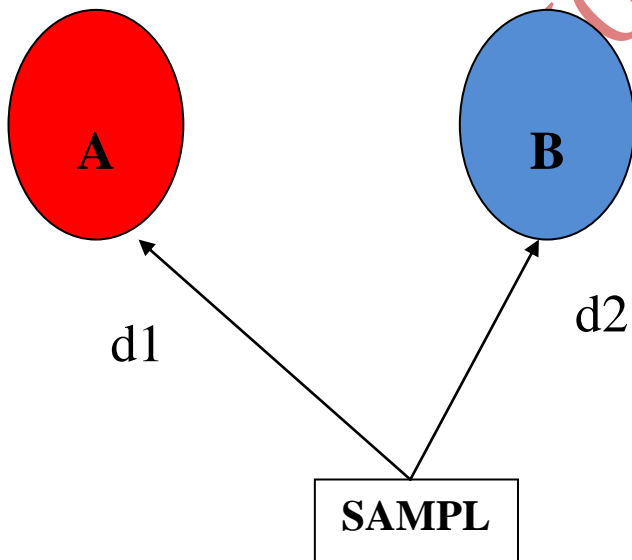


Fig 2: KNN Approach

The KNN evaluated the distance d_1 and d_2 and classifies the sample to belong to a dataset based on

$$z = \min(d_1, d_2) \quad (7)$$

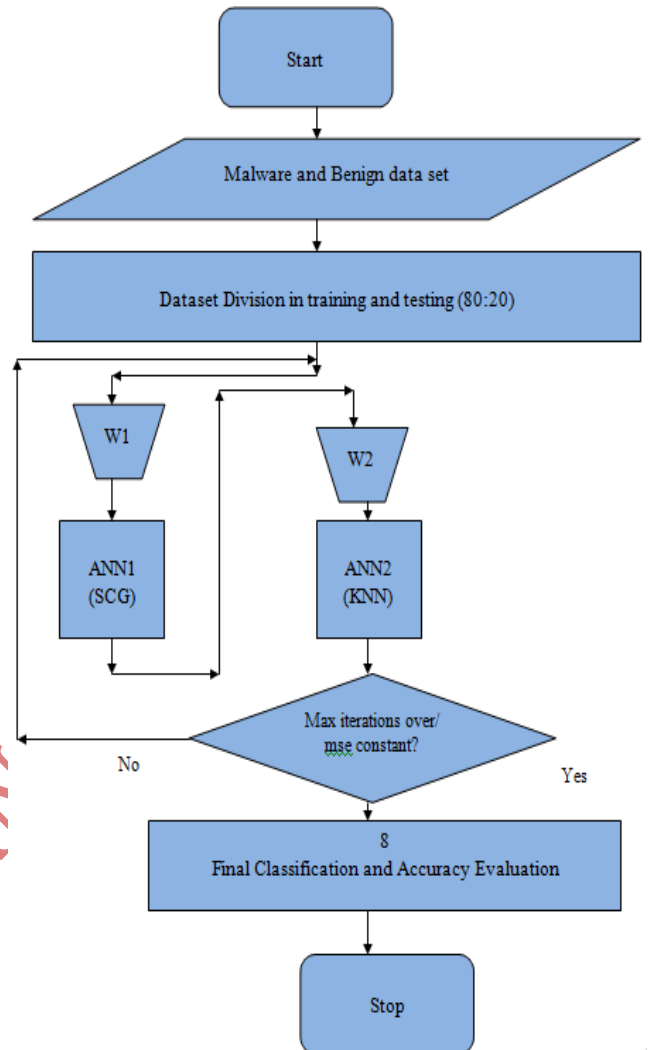


Fig 2: Flowchart for proposed approach

The proposed approach uses an **ada-boost** approach for android malware detection. In this approach, the output of one neural network is fed as the input to another neural network. The characteristic of such an approach is the fact that it can achieve higher effectiveness of classification accuracy compared to a single neural architecture for classification since the parameters which distinguish malwares and non-malwares are very similar and often makes the classification accuracy plummet.

5. RESULTS

The system has been simulated on MATLAB 2018a.

The evaluation parameters are:

- a) Classification accuracy
- b) Confusion matrix
- c) Neural network training parameters.

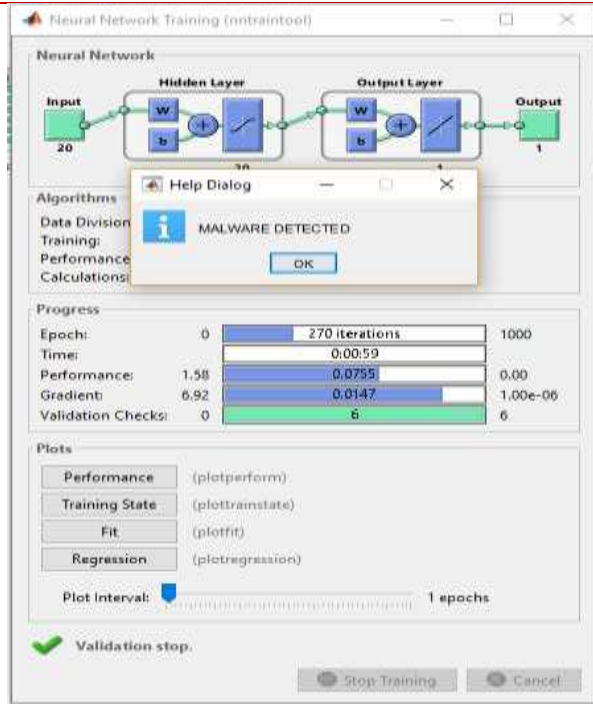


Fig 3: Malware Classification by System with training parameters

The figure above depicts the designed neural network for the classification. The training algorithm used is the scaled conjugate gradient (SCG) approach.

The figure above depicts the variation of the gradient and validations as per the variations in the number of iterations.

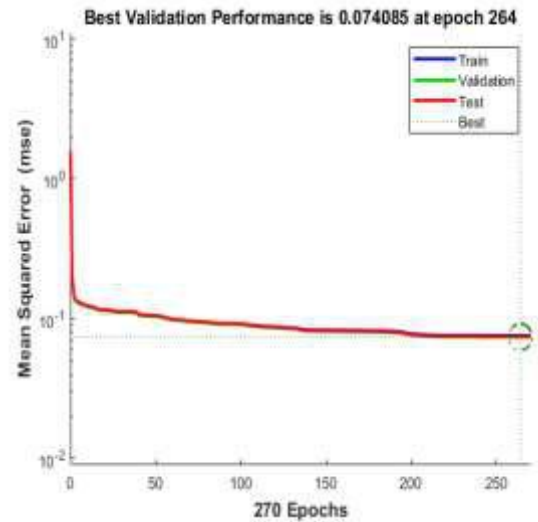


Fig 5: Variation of gradient and validations with respect to epochs.

The figure above depicts the variation of the mean square error as per the variations in the number of iterations.

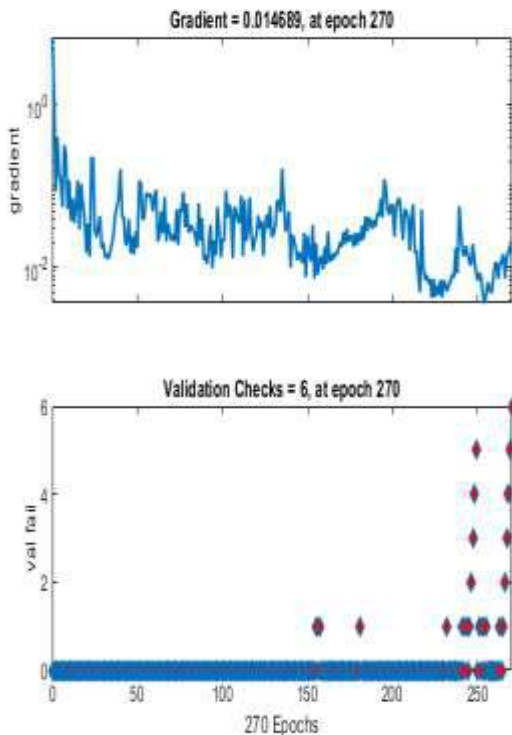


Fig 4: Variation of gradient and validations with respect to epochs.

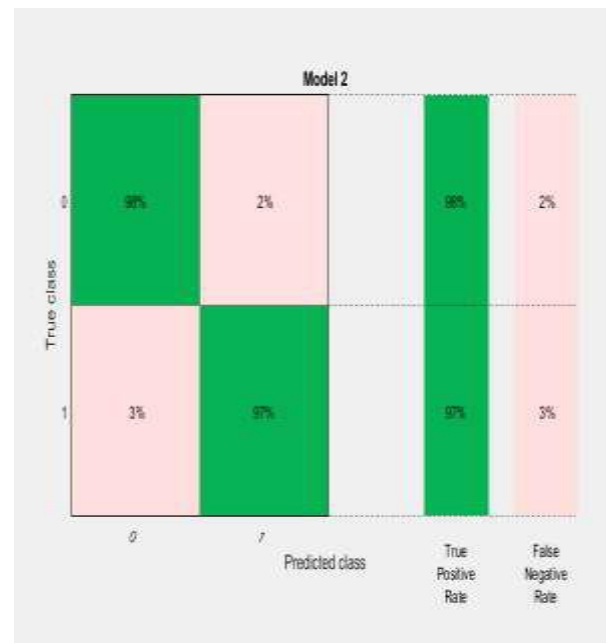


Fig 6: Confusion matrix for KNN approach.

The figure above depicts the confusion matrix for the used data set.

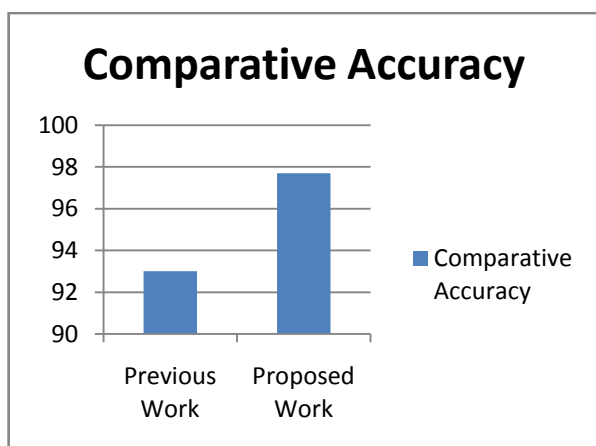


Fig 7: Comparative Accuracy Analysis w.r.t. previous work [1]

The figure above depicts the comparative accuracy attained by proposed work and previous work.

6. CONCLUSION

It can be concluded from previous discussions that the rise in malwares for android operating system have been on a high and detecting malwares with high accuracy is challenging. The proposed approach uses an **ada-boost** approach for android malware detection. In this approach, the output of one neural network is fed as the input to another neural network. The characteristic of such an approach is the fact that it can achieve higher effectiveness of classification accuracy compared to a single neural architecture. It has been shown that the proposed system achieves higher accuracy compared to previously existing work.

7. REFERENCES

- [1] Hossein Sayadi, Nisarg Patel, Sai Manoj P D, Avesta Sasan, Setareh Rafatirad, Houman Homayoun, "Ensemble Learning for Effective Run-Time Hardware-Based Malware Detection: A Comprehensive Analysis and Classification", IEEE 2018.
- [2] ElMouatez Billah Karbab, Mourad Debbab, Abdelouahid Derhab, Djedjiga Mouheb, "MalDozer: Automatic framework for android malware detection using deep learning", ELSEVIER 2018.

[3] Shifu Hou, Aaron Saas, Lingwei Chen, Yanfang Ye, Thirimachos Bourlai, "Deep Neural Networks for Automatic Android Malware Detection", ACM 2017.

[4] R. Vinayakumar, K. P. Soman, Prabaharan Poornachandran, "Deep android malware detection and classification", IEEE 2017.

[5] Sergei Bezobrazov, Anatoly Sachenko, Myroslav Komar, Vladimir Rubanau, "The Methods Of Artificial Intelligence For Malicious Applications Detection In Android OS", Ijc 2016.

[6] Konstantinos Demertzis and Lazaros Iliadis, "Bio-inspired Hybrid Intelligent Method for Detecting Android Malware", Springer 2016.

[7] Harry Kurniawan, Yusep Rosmansyah, Budiman Dabarsyah, "Android anomaly detection system using machine learning classification", IEEE 2015.

[8] Blas Torregrosa, "A framework for detection of malicious software in Android handheld systems using Machine Learning techniques", Semantic Scholar 2015.

[9] Wei Yu, Linqiang Ge, Guobin Xu, Xinwen Fu, "Towards Neural Network Based Malware Detection on Android Mobile Devices", Springer 2014.

[10] Mohd Zaki Mas'ud, Shahrin Sahib, Mohd Faizal Abdollah, Siti Rahayu Selamat, Robiah Yusof, "Ensemble Learning for Effective Run-Time Hardware-Based Malware Detection: A Comprehensive Analysis and Classification", IEEE 2014.

[11] Mo Ghorbanzadeh, Yang Chen, Zhongmin Ma, T. Charles Clancy, Robert McGwier, "A neural network approach to category validation of Android applications", IEEE 2013.

[12] John Demme, Maycock, Jared Schmitz, Adrian Tang, Adam Waksman, Simha Sethumadhavan, Salvatore Stolfo, "On the feasibility of online malware detection with performance counters", ACM 2013.

[13] Omer H. Abdelrahman, Erol Gelenbe, Gökçe Görbil Boris Oklander, "Mobile Network Anomaly Detection and Mitigation: The NEMESYS Approach", IEEE 2013.

[14] Te-En Wei, Ching-Hao Mao, Albert B. Jeng, Hahn-Ming Lee, Horng-Tzer Wang, Dong-Jie Wu, "Android Malware Detection via a Latent Network Behavior Analysis", IEEE 2012.

[15] Borja Sanz, Igor Santos, Carlos Laorden, Xabier Ugarte-Pedrero, Pablo Garcia Bringas, "On the automatic categorisation of android applications", IEEE 2012.

[16] Min Zhao, Fangbin Ge, Tao Zhang, Zhijian Yuan, "AntiMalDroid: An Efficient SVM-Based Malware Detection Framework for Android", Springer 2011.

[17] Peter Teufl, Stefan Kraxberger, Clemens Orthacker, Günther Lackner, Michael Gissing, Alexander Marsalek, Johannes Leibetseder, Oliver Prevenhieber, "Android Market Analysis with Activation Patterns", Springer 2011.

ijournals