Deep Learning in Drebin: Android malware Image Texture Median Filter Analysis and Detection

Luo Shi-qi[1*], Ni Bo[1], Jiang Ping[1], Tian Sheng-wei[2], Yu Long[2], Wang Rui-jin[3]

[1] School of Computer, Hubei Polytechnic University,
No.16, Guiling NorthRoad, XialuDistrict, Huangshi, Hubei 435003 - P.R. China
[e-mail: sq930911@hbpu.edu.cn]

[2] School of Information Science and Engineering, Xinjiang University
No.666, Shengli Road, Tianshan District, Urumqi, Xinjiang 830046 - P.R. China
[e-mail: tianshengwei@163.com]

[3] School of Computer Science and Engineering, University of Electronic Science and Technology of China
Chengdu, Sichuan, 610054 - P.R. China
[e-mail: ruijinwang@uestc.edu.cn]

*Corresponding author: Luo Shi-qi

Received November 21, 2018; revised January 23, 2018; accepted February 12, 2019; published July 31, 2019

Abstract

This paper proposes an Image Texture Median Filter (ITMF) to analyze and detect Android malware on Drebin datasets[1]. We design a model of "ITMF" combined with Image Processing of Median Filter (MF) to reflect the similarity of the malware binary file block. At the same time, using the MAEVS (Malware Activity Embedding in Vector Space) to reflect the potential dynamic activity of malware. In order to ensure the improvement of the classification accuracy, the above-mentioned features(ITMF feature and MAEVS feature)are studied to train Restricted Boltzmann Machine (RBM) and Back Propagation (BP). The experimental results show that the model has an average accuracy rate of 95.43%

[1] http://www.sec.cs.tu-bs.de/~danarp/drebin/

This work is partially supported by the National Natural Science Foundation of China(61841301). The Ministry of education of Humanities and Social Science project(17YJAZH043). The Scientific Research Innovation Project of Education Innovation Plan for Graduate Students in Xinjiang Uygur Autonomous Region(XJGRI2017007). The Science the Technology Talent Training Project of Xinjiang Uygur Autonomous Region(QN2016YX0051). The Cernet Next Generation Internet Technology Innovation Project(NGII20170420). The Natural Science Project of Hubei Provience(2018CFB456,2017CFB745). The Research Project of Hubei Provincial Department of Education (Q20184504). Guiding Project of the Science and Technology Program of the Hubei Provincial Department of Education(B2018251).

http://doi.org/10.3837/tiis.2019.07.018 **ISSN : 1976-7277**

with few false alarms. to Android malicious code, which is significantly higher than 95.2% of without ITMF, 93.8% of shallow machine learning model SVM, 94.8% of KNN, 94.6% of ANN.

Keywords: malware, Image Texture Median Filter, Malware Activity Embedding in Vector Space

1. Introduction

With the rapid development of Internet Technology, malware detection has become the difficulty and focus of Intrusion Detection System(IDS) in Network Space Security(NSS). Driven by economic benefits and anti-detection technologies, the number of malware has grown exponentially.[1][2] At the same time, Varieties of malware are emerging in an endless stream, which leads to a rising trend in security threat events. In May 2017, a computer ransomware named WannaCry is spreading worldwide, and has infected more than 100 countries. The most serious areas are concentrated in the America, Europe, Australia, China is also influenced. Many colleges have been infected and spread to large public service areas seriously such as airports, customs, and public security networks. Android is very popular in the field of mobile terminals, which occupies a large market share of mobile terminals. With the widespread use of Android mobile phones, Android-based malware has also developed rapidly. How to detect Android malware is particularly critical. At present, there have been many research progresses in Android malware analysis abroad. Representative researches include static analysis, dynamic analysis, and machine learning method.

However, the analysis and detection techniques based on shallow machine learning methods are mostly based on shallow models. Which have simple functions in the modeling process and limited ability in the expression of complex functions and classification problems and generalization ability. The classification accuracy are not high.

Aiming at problem of low detection and classification accuracy in detection of classification, this paper uses the method of deep learning to analysis and detect malware.

It is a proposal that using malware texture fingerprint to express the similarity of the content of the malware binary file block. At the same time, to improve the detection accuracy, the ITMF is introduced to reflect the similarity of the malware, and the MAEVS is used to reflect the malware potentially dynamic activities. Then two types of features have been merged together. Under the Android mobile terminal, the concept of deep learning have been applied to solve the problems of malware feature extraction, identification, and detection.

2. Related work

The analysis and detection model of malware usually comprises of two pieces: feature extraction and classification. The current feature extraction method is usually seperated into servel types: static analysis[3], dynamic analysis[4], static and dynamic fused analysis[5], the graphs-based approach[6][7][8][9][10] et.

1) Malware static analysis methods

The static analysis methods mainly include: signature-based, code semantic based, heuristic scanning techniques.

The signature-based detection[11]: the signature-based detection is a widely used approach in malware analysis. According to this method, the binary executable files are transformed to represent hashes which are matched with a database of known malware samples. Most commercial antivirus software makes use of this mechanism. Such as Norton, McAfee, and Kaspersky etc. At this stage, these techniques are mainly based on noise guidance and automatic generation distribution. Therefore early malware samples are shorter and the morphology is single, this detection method has achieved good results as long as there is a signature of this malware in the signature database. Malware analysis platforms are constantly evolving, increasingly malware such as viruses, Trojans, and worms also seriously affect people's lives. And to evade detection, malware anti-analysis techniques (such as packing, code obfuscation, information hiding Technology) are also constantly improving its ability. This will be a big challenge for detection and analysis.

The code-semantics-based[12]: the code-semantics-based analyzes the meaning of the instructions. In addition, it obtains the flow chart and the functional block diagram of malware, and to determine whether the program is malicious, it analyzes the functions and intentions of the program.

The heuristic scanning technique[13]: the heuristic scanning technique is actually an improvement based on the signature detection method. The main idea of this method is: when extracting the features of the file to be detected, it is compared with the characteristics of the signature library's known malware. As long as the match reaches a given threshold, the file is deemed to contain malware.

Kernel functions analysis: Generally, some kernel functions are called when the malware is executed, and the calling code of the malware has a great difference from benign. Using this principle, when scanning a program, you can extract kernel functions from it and the frequency, and compare it with the known code of the kernel functions in the code library. This method not only effectively detects known malware but also recognizes variants and unknown malware. However, it is difficult to capture the dynamic characteristics in the static analysis of malware, besides, it lacks of monitoring of program behavior.

2) Dynamic analysis methods

The dynamic analysis methods of malware (such as active defense technology[14] and cloud killing technology[15]) have been used by more security vendors with the

development of anti-malware technology. Dynamic(also name behavioral analysis), which can monitor the behavior of applications at run-time. It performed by observing the behavior of the malware while it is actually running on a host system. Such as, TaintDroid, DroidRanger and DroidScope are dynamic analysis method. But these algorithms spot whole malicious activities on the smart phone, which involve millions of smartphones at large scale in practice and takes a lot of spends and time. At present, the dynamic analysis method monitors the system function at the system application layer and lacks the detection of memory and registers. It is difficult to detect the kernel-level malware, and it is also a hard task to ensure the integrity of the analysis.

3)Graphs-based approach

Graphs-based approach mainly contains control-flow graphs, data dependency graphs, permission event graphs, automated Behavioral Graph Matching. While, these graphs are checked against manually-crafted specifications to detect malware. Besides, these detectors tend to seek an exact match for a given specification and therefore can potentially be evaded by polymorphic variants. Furthermore, the specifications used for detection are produced from known malware families and cannot be used to battle zero-day malware.

The malicious code model of classification based on feature extraction usually use data mining techniques or machine learning methods. The malware classification model based on machine learning usually uses shallow machines learning models (such as SVM[16] , Naive Bayes[17], decision tree[18][19], Random Forest[20] , KNN[21]) or shallow machine learning models fused. Contrast to basic learning method for shallow structure algorithm, the limitation lies in limited samples and cell cases of complex function said ability is limited, its generalization ability for complex classification problems under certain constraints. Deep learning method plays a better generalization performance of the classification and can learn more about cell cases of complex function.

(1) The traditional methods (such as the signature-based detection, feature matching) can not be able to achieve a good recognition and classification effect for exponentially increasing malware, and they rely on manual operations, formulate rules, which cannot fully extract malicious characteristics of the code. This paper will solve the exponential growth of malware in automation and accuracy.

(2) Analyzing the malware using static analysis is a widely used technique, the static analysis does not need to execute the code, while, it can not cope with the Shell code, polymorphism, metamorphism etc. Besides, the dynamic analysis avoids the drawbacks above, but it needs to execute the malware in run-time to avoid them, the dynamic analysis of malware detection monitors malicious activity on the smart phone, it involved millions of smart phones in practice Paranoid Android at large scale is technically not feasible. This paper will use the combination of static and dynamic for detection and analysis, which can show their own advantages.

(3) By contrast basic learning method for shallow structure algorithm, the limitation lies in limited samples and cell cases of complex function said ability is limited, its generalization ability for complex classification problems under certain constraints. Deep learning method plays a better generalization performance of the classification and can learn more about cell cases of complex function.[22]

In our previous study, we have put forward malware image texture for analysis[23][24][25][26]. For improving accuracy of detection, in this paper, the ITMF and MAEVS are applied on feature extraction and classification of input data.

3. Method

3.1 Image Texture Median Filter

3.1.1 Median Filter

The MF is a nonlinear digital filtering technique, often used to remove noise from an image or signal. Such noise reduction is a typical pre-processing step to improve the results of later processing (for example, edge detection on an image). MF is very widely used in digital image processing, under certain conditions, it preserves edges while removing noise (but see discussion below), also having applications in signal processing.

Let X and Y respectively represent an image and its median-filtered image. The MF operation with $w \times w$ as the neighborhood is performed according to equation.

$Y_{(i,j)} = \mathbf{median}\{X_{(i+h,j+v)}\}$, In the formula $X_{(i+h,j+v)}$, $Y_{(i,j)}$ denotes X, Y at $(i+h,j+v)$, (i,j) the pixel value, where $h,v \in [\frac{1-w}{2}, \frac{w-1}{2}]$.

3.1.2 Image Texture Median Filter

It is a useful technique to used visualization in computer forensics, NSS, image analysis, image classification and large-scale image search, to name a few.

Recently, image texture-based classification was used to classify malware. Image texture is a block of pixels which contains variations of intensities arising from repeated patterns, and then analyze it with the idea of median filtering.

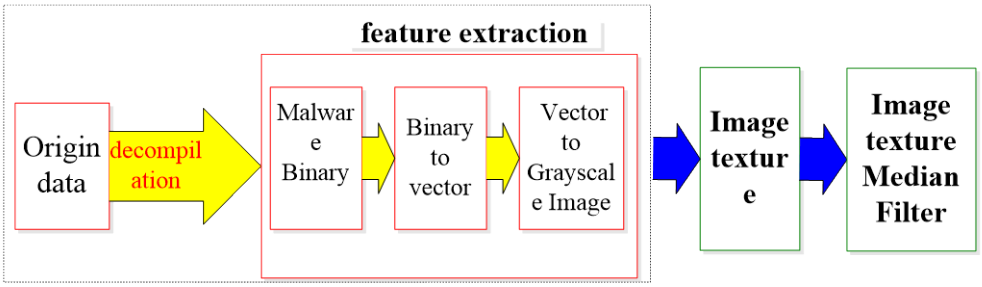


Fig. 1. Feature Extraction(Image Texture Median Filter)

Assume that there is a malware sets $M=\{m_i\}, i\in\{1,2,3,,,,n\}$, where $\boldsymbol{n}$ is the number of malware.

For malware m_i, it is mapped to an uncompressed grayscale image $Matrix(m_i)$, and the gray value of the image is stored as a matrix.

$$Matrix(m_i)=\begin{pmatrix} m_{i(1)} & m_{i(2)} & , & , & m_{i(k)} \\ m_{i(k+1)} & , & , & , & m_{i(2k)} \\ m_{i(2k+1)} & , & , & , & m_{i(3k)} \\ , & , & , & , & , \\ m_{i(nk+1)} & , & , & , & m_{i(nk+k)} \end{pmatrix}$$

Fig. 2. Image Texture Median Filter ***Matrix***

Based on the texture image median filter, first define a one-dimensional array N, $N=\{m_{i(1)},m_{i(2)},,,m_{i(k)},m_{i(k+1)},,,m_{i(nk+k)}\}$. Texture image median filtering $Q=\{q_j\}$, for $q_j=\text{median}(\sum_{o=j}^{j+r-1}m_{i(o)})$, where r is the median filter sliding window size.

3.2 MAEVS

The Android malware API call usually reflects the dynamic activity of a specific pattern software. For example, the malware sending a SMS will call SEND_SMS as a license, and using the dialing function will call android.hardware.telephony.

Analogical to the concept of word vectors in natural language processing, this paper suggestes MAEVS and maps malware to vector space.

$$\chi(x)=\begin{Bmatrix} 0 \\ 1 \\ 0 \\ n \\ l \end{Bmatrix}\begin{matrix} API\quad calls \\ Used\quad pemission \\ Intenstion \\ Url \\ Activity \end{matrix}$$

Fig. 3. MAEVS

In this paper, the five most common malware activity: API calls, Used permissions, Url, Intenstion, and Activity, are selected and mapped into the MAEVS.

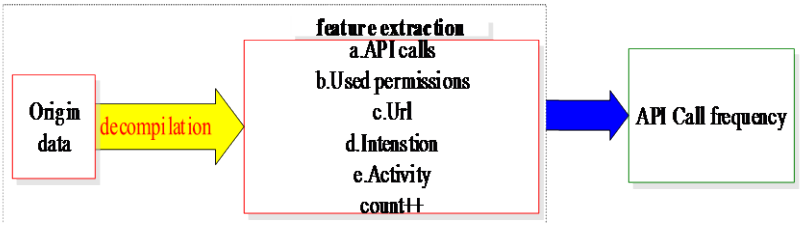


Fig. 4. Feature Extraction(MAEVS)

Furthermore, to raise the accuracy of classification algorithm on feature selection, on the basis of that, we amplify the implicit features of texture image median filter and API call in malware, to train Deep Belief Network .

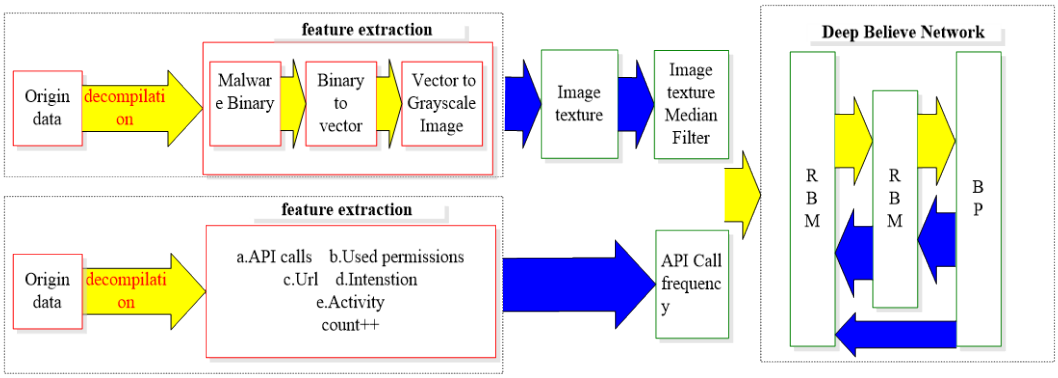


Fig. 5. Android malware Image Texture Median Filter Analysis based on DBN

3.3 DBN(Deep Belief Network)

3.3.1 RBM(Restricted Boltzmann Machine)

Restricted Boltzmann Machine (RBM) is a undirected graphical model. There are no links between units of the same layer, only between input (or visible) units x_j and hidden (also invisible) units h_i. The difference between standard Boltzmann machines and RBM is that in the restricted model units within the same layer are not connected. which makes inference and learning within this graphical model tractable.

3.3.2 BP(back propagation)

BP(back propagation) is a common method of training artificial neural networks and used in conjunction with an optimization method such as gradient descent. The algorithm repeats a two phase cycle, propagation and weight update. When an input vector is presented to the network, it is propagated forward through the network, layer by layer, until it reaches the output layer. The output of the network is then compared to the desired output, using a loss function, and an error value is calculated for each of the neurons in the output layer. The

error values are then propagated backwards, starting from the output, until each neuron has an associated error value which roughly represents its contribution to the original output.

3.3.3 CD algorithm

The goal of RBM training is to make marginal probability distribution $p(v)$ fit probability distribution of training samples based on justifying the parameters of model. To achieve this, we use k-steps contrastive divergence learning algorithm to train RBMs which is a standard way to train RBMs. The idea of CD-k is quite simple: the chain is run for only k steps, starting from an example $v^{(0)}$ of the training set and yielding the sample $v^{(k)}$. Each step t consists of sampling $h(t)$ from $p(h \mid v^{(t)})$ and sampling $v^{(t+1)}$ from $p(v \mid h^{(t)})$ subsequently. The gradient in equation (2) with respect to θ of the log-likelihood for one training pattern $v^{(0)}$ is then approximated by equation (1),

$$CD_k(\theta, v^{(0)}) = -\sum_h p(h \mid v^{(0)}) \frac{\partial \varepsilon(v^{(0)}, h)}{\partial \theta} + \sum_h p(h \mid v^{(k)}) \frac{\partial \varepsilon(v^{(k)}, h)}{\partial \theta} \quad (1)$$

In the following, we restrict our considerations to RBMs with binary units for which, which is showed in equation (2).

$$E_{p(h_i \mid v)}[h_i] = sigmoid(c_i + \sum_{j=1}^{m} w_{ij} v_j) \quad (2)$$

3.3.4 DBN

Hinton, Osindero, and Teh introduced a greedy layer-wise unsupervised learning algorithm for Deep Belief Networks (DBN) in 2006, shows in **Fig. 6** The training strategy for such networks may hold great promise as a principle to help address the problem of training deep networks. Upper layers of a DBN are supposed to represent more "abstract" concepts that explain the input data whereas lower layers extract "low-level features" from the data. As an unsupervised learning in deep architectures, DBN is a multi-layered probabilistic generative model. Deep Belief Network can be defined as a stack of Restricted Boltzmann machines with a Back Propagation(BP) to fine tuning. Previously, Deep Belief networks has been successfully employed in recognize, cluster and generate images, video sequences and motion-capture data.

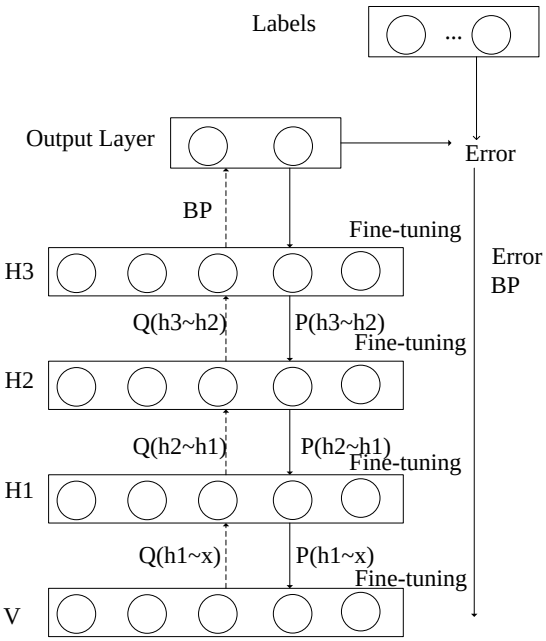


Fig. 6. DBN

4. Experiment

Drebin(the malware dataset) contains 5,560 applications from 179 types different malware families. The samples have been collected in the period of August 2010 to October 2012 and were made available to us by the Mobile-Sandbox project. Besides, we also collected 123453 kinds of benign application for research.

In order to guarantee the efficient stringency of experiments data applied to the model, this paper selects 1550, 2620, 5825 samples from the dataset.

In the experiment, it is crucial to create an appropriate experiment environment for detection and analysis.

Table 1. Experimental environment

Parameter	Value
OS	Windows 7 64 bit platform
CPU	Intel i7-7700@3.6HZ 3.6GHZ
GPU	NVIDA GeForce GTX 1080
RAM	32G
Hard disk	120G SSD+4T HDD
CUDA	7.5
CUDNN	5.0
Python	2.7.3

It is essential to constructs an experiment environment of Python 2.7.3 with suitable and available module. It is listed in the following table.

Table 2. Python module for experiment

name	introduction	version
numpy	The fundamental package for scientific computing with Python	1.9.2
pandas	Data structures and data analysis tools	0.16.0rc1(0.11.1)
PIL	Python Imaging Library	1.1.7
Scikit-learn	Python module for machine learning	0.16.1
scipy	Python-based ecosystem of open-source software for mathematics	0.15.1
twisted	An asynchronous networking framework	15.4.0
six	Compatibility library	1.7.3
pip	The PyPA recommended tool for installing Python packages.	8.1.2
wheel	A built-package format for Python	0.29.0
datutil	Extensions to the standard Python datetime module	2.2
pyparsing	A general parsing module for Python	2.0.1
setuptools	Package development process library	0.16.1
pytz	Brings the Olson tz database into Python	2016.6.1
nolearn	Python maching learning module	0.6.0
theano	Python deep learning module	0.8.2
gdbn	Python deep learning DBN moudle	0.2
keras	Python deep learning module	2.0.8
Tensorflow-GPU	Python deep learning module	1.4.0

This experiment has selected different numbers data samples to determine variable parameters (feature number, number of network layers, etc.), with 1550, 2620, 5825 experimental data, in addition, 60% of the samples are training data, and the other 40% are test data. At the same time, a unified statistical indicator is used to evaluate the performance of the model, and the total prediction accuracy (Q) formula (3) is used to measure the performance of the model.

$$Q= (TP+TN)/(TP+TN+FP+FN) \quad (3)$$

In equation (3), TP represents the number of true positives, FP represents the number of false positives, TN represents the number of true negatives, and FN represents the number of false negatives.

4.1 Effect of variable parameters on the experimental results in the model

The variable parameters in the experiment include the number of features, the number of network layers, etc. In order to optimize the experimental results, the variable parameters are tested on the same data set, and the optimal parameters are selected. The experimental results are shown in **Fig. 7** and **Table 3**.

Table 3. Pix number of image texture on experimental results

	500	1000	2000	2500
1550	94.0	94.1	94.8	94.3
2620	94.08	94.9	95.1	94.6

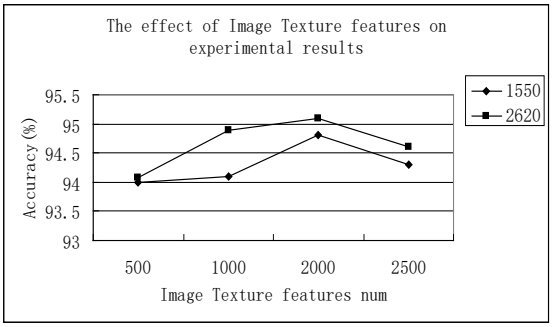


Fig. 7. Pix number of image texture on experimental results

Table 4. The effect of the number of DBN layers on the experimental results

Layer	1550	2620
2	94.8	95.1
3	94.6	94.8
4	94.1	94.0

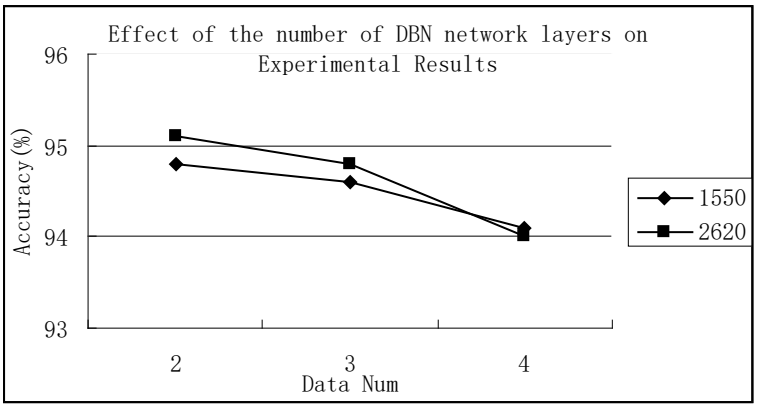


Fig. 8. The effect of the number of DBN layers on the experimental results

Based on the results of the above groups of experiments, it is clear to know: when the number of texture fingerprint features is 2000, the classification accuracy is the highest. When the number of DBN network layers is 2, the classification accuracy is the highest. Therefore, the parameters of the model in the experiment are listed in **Table 5**.

Table 5. Parameters of the model

parameters	values
Image feature num	2000
DBN layer	2
optimize	BP
test_size	0.4
epochs	300
r	9

4.2 Effect of Malware Image Texture(MIT) Features on Experimental Results

In order to ensure that the MIT feature has an important influence on the experimental results, we use the MIT feature to fuse the MAEVS and only the MAEVS on the basis of the aforementioned parameter model. The experimental results are shown in **Fig. 9** and **Table 6**.

Table 6. Effect of MIT Features on Experimental Results

Data num	MIT+MAEVS	MAEVS only
1550	94.8	94.6
2620	95.1	95.0
5825	95.7	93.5

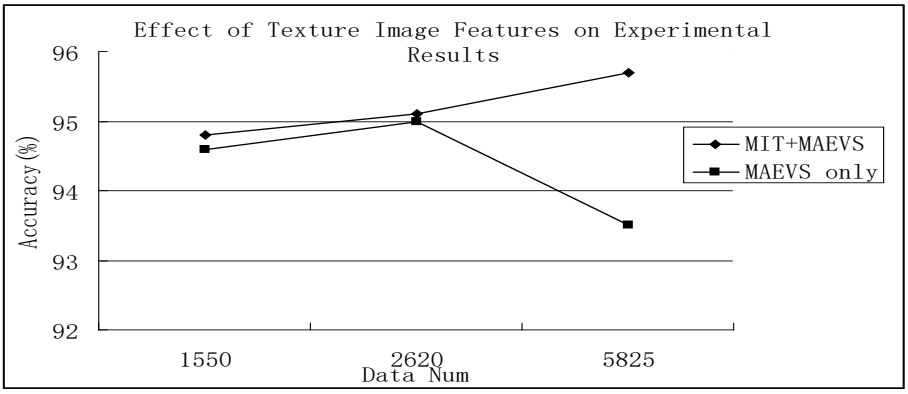


Fig. 9. Effect of Texture Image Features on Experimental Results

4.3 The Comparison of Introducing Malware ITMF

In order to demonstrate the feasibility of the proposed algorithm, the malware ITMF was introduced to compare experiments without malware ITMF. The experimental results are listed in **Fig. 10** and **Table 7**.

Table 7. Compare to without malware ITMF

Data num	DBN(MIT+ITMF+MAEVS)	DBN(MIT+MAEVS)
1550	95.0	94.8
2620	95.4	95.1
5825	95.9	95.7

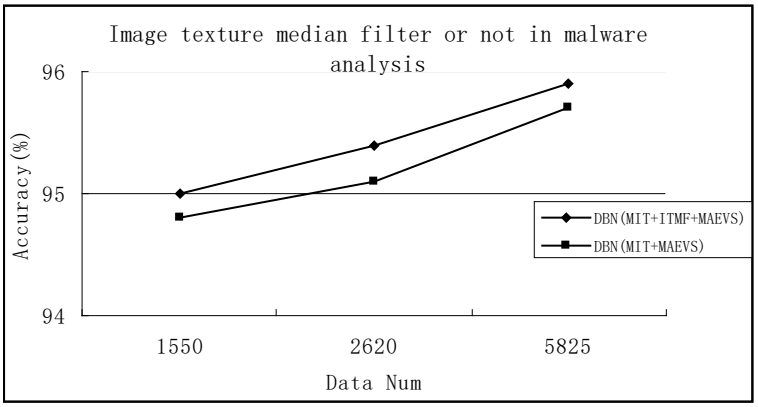


Fig. 10. Compare to without malware ITMF

4.4 Comparison with Shallow Machine Learning Model

In order to demonstrate the feasibility of this algorithm, the deep learning model is merged with the malware ITMF and the MAEVS. Compared with the shallow machine learning model SVM, KNN, and ANN, the results of classification accuracy are listed as follows.

Table 8. Comparison with shallow machine learning model

Data num	DBN (MIT+ITMF+MAEVS)	DBN (MIT+MAEVS)	SVM (MIT+MAEVS)	KNN (MIT+MAEVS)	ANN (MIT+MAEVS)
1550	95.0	94.8	92.7	94.5	93.6
2620	95.4	95.1	94.2	95.0	95.0
5825	95.9	95.7	94.5	95.1	95.2

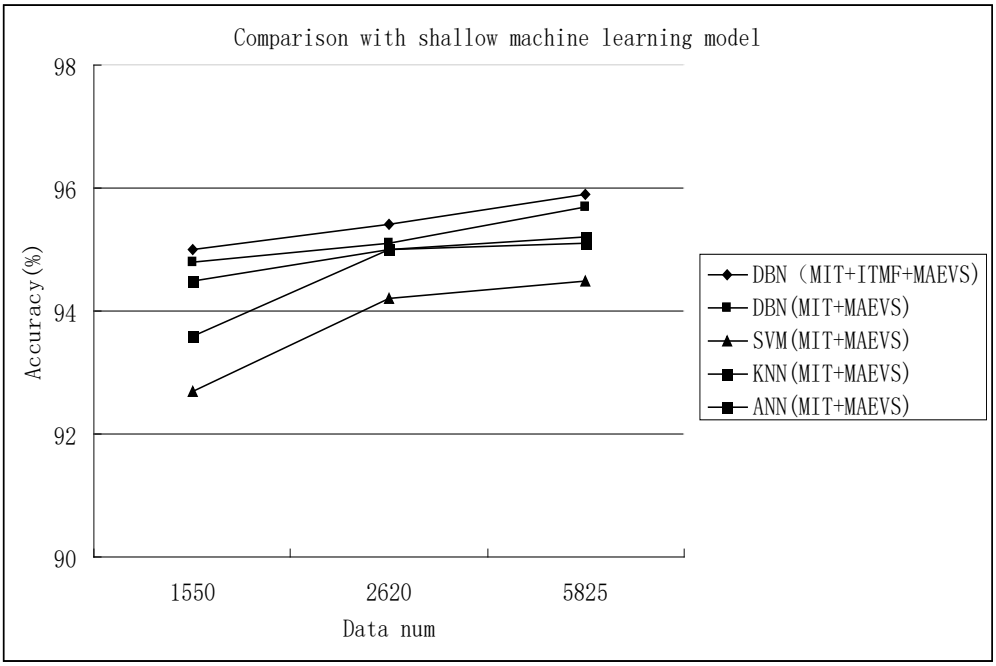


Fig. 11. Comparison with shallow machine learning model

5. Conclusion

This paper is different from previous researches. Based on the MAEVS, the malware ITMF has been proposed. The DBN model was used to extract features of the malware ITMF and the MAEVS.

From the **Fig. 10** and **Fig. 11**, a large number of experimental results showed great performance in accuracy. From the introduction of ITMF, the average classification accuracy for Android malware can reach 95.43%, which is significantly higher than 95.2% without malware ITMF, 93.8% for SVM, 94.8% for KNN, and 94.6% for ANN.

Acknowledgements

We would like to thank all the participants in our reasearch that provided useful and detailed feedback. Meanwhile, I would thank all my team and my school for the research.

This work is partially supported by the National Natural Science Foundation of China(61841301). The Ministry of education of Humanities and Social Science project(17YJAZH043). The Scientific Research Innovation Project of Education Innovation Plan for Graduate Students in Xinjiang Uygur Autonomous Region(XJGRI2017007). The Science the Technology Talent Training Project of Xinjiang Uygur Autonomous Region(QN2016YX0051). The Cernet Next Generation Internet Technology Innovation Project(NGII20170420). The Natural Science Project of Hubei

Provience(2018CFB456,2017CFB745). The Research Project of Hubei Provincial Department of Education (Q20184504). Guiding Project of the Science and Technology Program of the Hubei Provincial Department of Education(B2018251).

Reference

[1] Balaganesh D, Chakrabarti A, Midhunchakkaravarthy D, "Smart Devices Threats, Vulnerabilities and Malware Detection Approaches: A Survey," *EJERS*, 3(2), 7, 2018. Article (CrossRef Link)

[2] Li Jian,Wang Zheng et., "An Android Malware Detection System Based on Feature Fusion," *Chinese Journal of Electronics*, 27(6), 1206-1213, 2018. Article (CrossRef Link)

[3] Seshagiri P, Vazhayil A,Sriram P, "AMA: Static Code Analysis of Web Page for the Detection of Malicious Scripts ," *Procedia Computer Science*, 93, 768-773, 2016. Article (CrossRef Link)

[4] Willems C, Holz T, Freiling F, "Toward Automated Dynamic Malware Analysis Using CWSandbox," *IEEE Security & Privacy Magazine*, 5(2), 32-39, 2007. Article (CrossRef Link)

[5] Elhadi A A E, Maarof M A, Osman A H, "Malware detection based on hybrid signature behaviour application programming interface call graph," *American Journal of Applied Sciences*, 9(3), 283-288, 2012. Article (CrossRef Link)

[6] Park Y, Reeves D, Mulukutla V, et al., "Fast malware classification by automated behavioral graph matching," in *Proc. of CSIIRW '10 Proceedings of the Sixth Annual Workshop on CyberSecurity and Information Intelligence Research*, 1-4, 2010. Article (CrossRef Link)

[7] Christodorescu M, Jha S, Seshia S A, et al., "Semantics-aware malware detection," in *Proc. of Security and Privacy, 2005 IEEE Symposium on*, 32-46, 2005. Article (CrossRef Link)

[8] Fredrikson M, Jha S, Christodorescu M, et al., "Synthesizing Near-Optimal Malware Specifications from Suspicious Behaviors," in *Proc. of IEEE Symposium on Security and Privacy. IEEE Computer Society*, 45-60, 2010. Article (CrossRef Link)

[9] Kolbitsch C, Comparetti P M, Kruegel C, et al., "Effective and efficient malware detection at the end host," in *Proc. of 18th Usenix Security Symposium*, 351-366, Montreal, Canada, August10-14, 2009. Article (CrossRef Link)

[10] Chen K Z, Johnson N, D'Silva V, et al., "Contextual Policy Enforcement in Android Applications with Permission Event Graphs," *Heredity*, 110(6), 586, 2013. Article (CrossRef Link)

[11] Kruegel C, Toth T, "Using Decision Trees to Improve Signature-Based Intrusion Detection," *Lecture Notes in Computer Science*, 2820, 173-191, 2003. Article (CrossRef Link)

[12] Venkitaraman R, Gupta G, "Static program analysis of embedded executable assembly code," in *Proc. of International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, CASES 2004, Washington Dc, Usa*, 157-166, 2004. Article (CrossRef Link)

[13] Zhang B, Li Q, Ma Y, "Research on dynamic heuristic scanning technique and the application of the malicious code detection model," *Information Processing Letters*, 117, 19-24, 2017. Article (CrossRef Link)

[14] Chen Y Q, Xiao-Ping W U, Fu Y, et al., "Active Defense strategies selection for network mixed malicious action," in *Proc. of International Workshop on Cloud Computing & Information Security*, 52(1391), 336-340, 2013. Article (CrossRef Link)

[15] Aijun Jiang, Zhifeng Liu, Qinglong Kong, Bo Zhang, Tong Yao, "Scanning device, cloud management device, method and system for checking and killing malicious programs," *US, US20150317478 A1*, 2015. Article (CrossRef Link)

[16] Arp D, Spreitzenbarth M, Hübner M, et al., "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket," in *Proc. of Network and Distributed System Security Symposium*, 2014. Article (CrossRef Link)

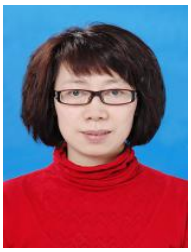
[17] Schultz M G, Eskin E, Zadok F, et al., "Data mining methods for detection of new malicious executables," in *Proc. of Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on. IEEE*, 38-49, 2001. Article (CrossRef Link)
[18] Zerina Mašetic, Subasi A , Azemovic J, "Malicious Web Sites Detection using C4.5 Decision Tree," in *Proc. of Iusscrg, Fourth Regional Conference on Soft Computing*, 2016. Article (CrossRef Link)
[19] Zhao G, Wang P, Wang X, et al., "The Detection Method for Two-dimensional Barcode Malicious URL Based on the Decision Tree," *Information Security & Technology*, 2014. Article (CrossRef Link)
[20] Alam M S, Vuong S T, "Random Forest Classification for Detecting Android Malware," *Green Computing and Communications. IEEE*, 663-669, 2013. Article (CrossRef Link)
[21] Liu Z, Juan D U, Zhian Y I, "Application of a Improved Categorization Algorithm in the Malicious Information Filtering," *Microcomputer Applications*, 2011.
[22] Bengio Y, "Learning deep architectures for AI," *Foundations and trends in machine learning*, 2(1), 1-127, 2009. Article (CrossRef Link)
[23] Luo Shiqi, Tian Shengwei, Yu Long, Yu Jiong and Sun Hua, "Android malicious code Classification using Deep Belief Network," *KSII Transactions on Internet and Information Systems*, 12(1), 454-475, 2018. Article (CrossRef Link)
[24] Luo Shiqi, Tian Shengwei,et., "Research on malicious code classification algorithm of stacked auto encoder," *Application Research of Computers*, 35(1), 261-265, 2018. Article (CrossRef Link)
[25] Luo Shiqi, Tian Shengwei ,et. "Research strategy of classify malicious code into families on the method of deep belief networks," *Journal of Chinese Computer Systems*, 38(11), 2465-2470, 2017 Article (CrossRef Link)
[26] Luo Shiqi, Tian Shengwei, Yu Long, Yu Jiong and Sun Hua, "Android malware detection based on texture fingerprint and malware activity vector space," *Journal of Computer Application*, 38(4), 1058-1063, 2018. Article (CrossRef Link)

Luo Shi-qi, born in Hubei of China in 1993. Master, Assistant in School of Computer, Hubei Polytechnic University. His main research interests include Information Security.

Ni Bo, born in Hubei of China in 1982. PHD, Associate professor in School of Computer, Hubei Polytechnic University. He received the PHD degree in Wuhan University. His main research interests include Computer Vision.

Jiang Ping, born in Hubei of China in 1983. PHD, lecturer in School of Computer, Hubei Polytechnic University. He received the PHD degree in Huazhong University of Science and Technology. His main research interests include Intelligence Technology.

Tian Sheng-wei, born in 1973. Professor and PhD supervisor in Xinjiang University. His main research interests include Intelligence Computing.



Yu Long, born in 1974. Professor and PhD supervisor in Xinjiang University. Her research interests include Intelligence Technology.

WANG Rui-jin, born in 1980. PhD, lecturer in School of Computer Science and Engineering, University of Electronic Science and Technology of China. His main research interests include Quantum Communication Security.