Android Malware Analysis and Detection Based on Attention-CNN-LSTM

Luo shiqi[1*], Liu Zhiyuan[1], Ni Bo[1], Wang Huanhuan[2], Sun Hua[2], Yuan Yong[1]
[1]School of Computer, Hubei Polytechnic University, Huangshi, Hubei 435003, P. R. China.
[2]School of Software, Xinjiang University, Urumqi, Xinjiang, 830008, P. R. China.

*Corresponding author. Tel.: (+86)15927466968; email: sq930911@hbpu.edu.cn
Manuscript submitted July 20, 2018; accepted September 13, 2018.
doi: 10.17706/jcp.14.1.31-43

Abstract: The increasing number of Android malware has made detection and analysis more difficult, aiming to the current malware attacking Android. This paper proposes an Android malware analysis and detection technology based on Attention-CNN-LSTM, which is a types of Multimodel Deep Learning. Selecting open source malware datasets of Drebin for research, extracting texture fingerprint information of Android malware to reflect the similarity of malware binary file blocks, at the same time, in order to improve the detection accuracy, AndroidMainfest.xml is treated as a text document, and its contextual text features are extracted through NLP. Besides, the above two types of features are merged to enhance the expression capability of texture fingerprint information , and Deep Belief Network is used to screen the above features. Above all, the texture fingerprint is processed by one-dimensional serial signal processing, and the end-to-end local correlation features are extracted according to a one-dimensional time-domain convolutional network. At the same time, considering the context relationship of the timing signal for the AndroidMainfest.xml text, combined with the LSTM model with stronger time-series modeling capabilities to analyze and detect the Android malicious code. The experimental results show that the proposed method can detect and analyze malware more effectively.

Keywords: Malware, Android, attention-CNN-LSTM, multimodel deep learning, Deep Belief Network (DBN).

1. Introduction

With the rapid development of the 3G/4G mobile Internet, a large number of malicious applications have emerged in the years [1], and they pose a great threat to the security of the Android platform [2]. The analysis and detection of Android malware has attracted widespread attention from academia and industry. How to improve the detection performance of malware [3], which classifiers should be selected, and which features should be used are key issues that need to be addressed urgently[4].

2. Related Work

The detection and analysis model of malicious code consists of two pieces: feature extraction and classification. The current feature extraction method is usually divided into serval types: static analysis [5], [6], dynamic analysis [7], [8], dynamic and static fused analysis [9]-[11], the graphs-based approach [12]-[14] et.

Malware static analysis methods mainly include: signature-based [15], [16], code semantic based [17], [18], heuristic scanning techniques [19]; Malware dynamic analysis methods mainly include: active defense [20] and cloud killing technologies [21].

The signature-based detection: 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 adopt this mechanism, such as Norton, McAfee, and Kaspersky et. 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, while 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: 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: 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. Based on 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.

Dynamic analysis methods: the dynamic analysis methods of malware (such as active defense technology and cloud killing technology) 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 performs by observing the behavior of the malware while it is actually running on a host system. For example, 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 cost 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.

Graphs-based approach: graphs-based approach mainly contains control-flow graphs, data dependency graphs, permission event graphs, automated Behavioral Graph Matching. However, 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.

In our previous study, we put forward image texture for analysis. [22]-[23] For improving accuracy of detection, in this paper, the multi-model deep learning, malware texture fingerprint and Malware Activity Embedding in Vector Space are applied on feature extraction and classification of data.

The main work and innovations of this paper are listed as follows:

1. This paper proposes Android malware texture fingerprint to reflect the similarity of binary file block content. Besides, it filters the malware activity vector space used to reflect the potential dynamic activity of malicious code. Finally, it combines the selected features listed before.

2. Multi-modal Deep Learning is used to Android malware detection. The texture fingerprint is processed by

one-dimensional sequence signal, and the end-to-end local correlation feature is extracted according to the one-dimensional time domain convolution network. At the same time, the contextual signal context relationship is considered aiming at AndroidMainfest.xml text, and the LSTM with strong time series modeling ability is used to analyze and detect Android malware, finally, the semantic information association between malicious code texts is strengthened.

3. Deep learning is applied to Android malware analysis and detection, using the classification algorithm to train the extracted feature sets, filtering out the effective features, and constructing the classifier based on Android malware through feature fusion, treating the constructed classifier test Android malware for detection and classification, which increased automation and accuracy of malicious code detection and classification.

3. Algorithm Model

3.1. Deep Belief Network

In 2006, Hinton *et al* introduced a greedy layer-wise unsupervised learning algorithm for Deep Belief Networks (DBN) [24], 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.

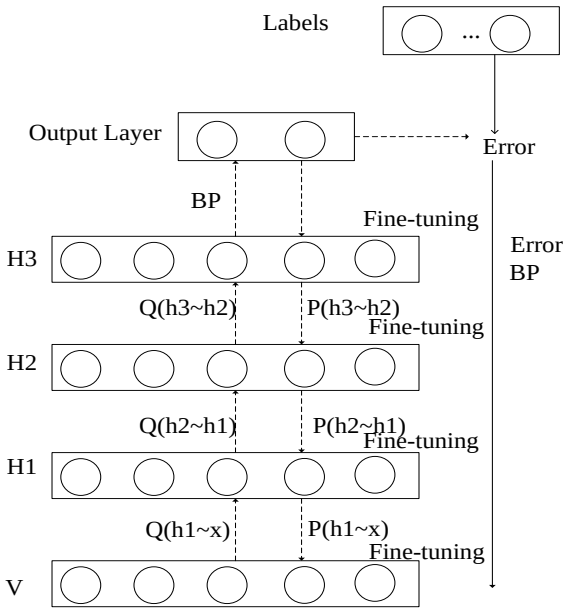

Fig. 1. DBN (Deep Belief Network).

Since the same type of malware has a certain similarity in the texture fingerprint, this paper proposes a malware fingerprint texture fingerprint feature extraction algorithm based on deep belief network. By combining image processing technology and malicious code detection technology, the malicious code is mapped to uncompressed grayscale image, and then the texture fingerprint feature is extracted using the Deep Believe Network.

Also, the Android 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 33 types Malware Activity Embedding in Vector Space and maps malware to vector space.

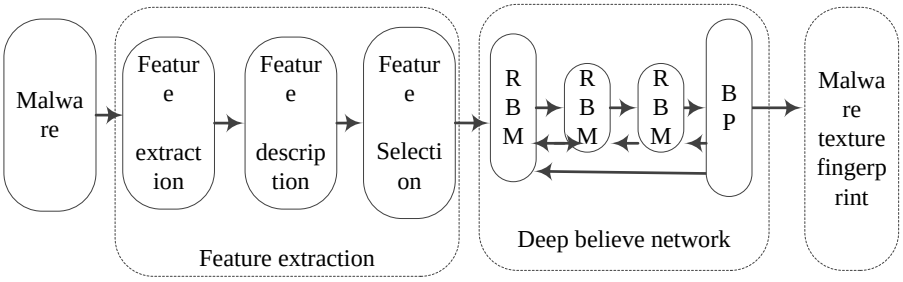

Fig. 2. Features of malware texture fingerprint analysis based on DBN.

$$
I(x)=\begin{pmatrix}
\text{android.hardware.telephony}\\
\text{android.permission.INTERNET}\\
\text{android.permission.READ\_PHONE\_STATE}\\
\text{android.permission.ACCESS\_CACH E\_FILESYST EM}\\
\text{android.permission.READ\_LOGS}\\
\text{android.hardware.screen.portrait}\\
\text{android.hardware.screen.landscape}\\
\text{SEND\_SMS}\\
\text{SendSMS}\\
\text{android.hardware.wifi}\\
\text{android.intent.action.MAIN}\\
\text{android.hardware.camera}\\
\text{android.intent.category.LAUNCHER}\\
\text{android.permission.WAKE\_LOCK}\\
\text{android.permission.VIBRATE}\\
\text{android.hardware.location.network}\\
\text{getSystemService}\\
\text{getSubscriberId}\\
\text{android.permission.RECEIVE\_BOO T\_COMPLETE D}\\
\text{android.permission.ACCESS\_NETW ORK\_STATE}\\
\text{DELETE\_PAC KAGES}\\
\text{WebView}\\
\text{ru.alpha.AlphaReceiver}\\
\text{android/telephony/TelephonyManager}\\
\text{BootReceiver}\\
\text{Obfuscation}\\
\text{android/content/Context}\\
\text{RestartAppTrigger}\\
\text{android.hardware.location.gps}\\
\text{android/media/AudioManager}\\
\text{android.permission.CALL\_PHONE}\\
\text{HttpPost}\\
\text{android/location/LocationManager}
\end{pmatrix}
$$

Fig. 3. Malware activity embedding in vector space.

And then the Malware Activity Embedding in Vector Space is extracted using the deep Belief network also.

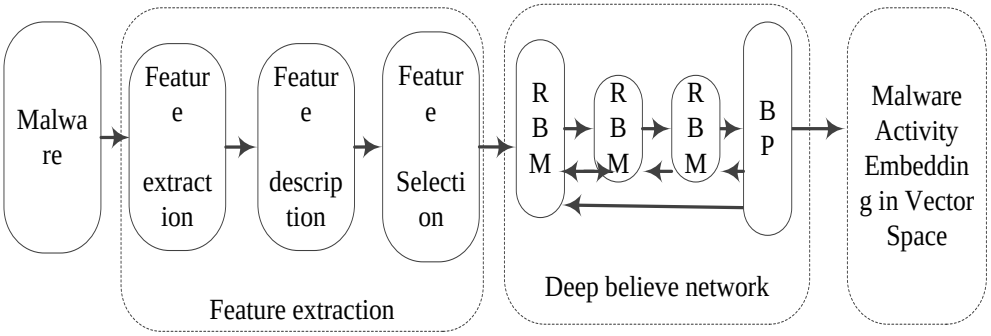

Fig. 4. Features of malware activity embedding in vector space analysis based on DBN.

3.2. Attention-CNN-LSTM

This paper proposes an Android malware analysis and detection technology based on Attention-CNN-LSTM, Attention-CNN-LSTM is consists of Attention, CNN and LSTM.

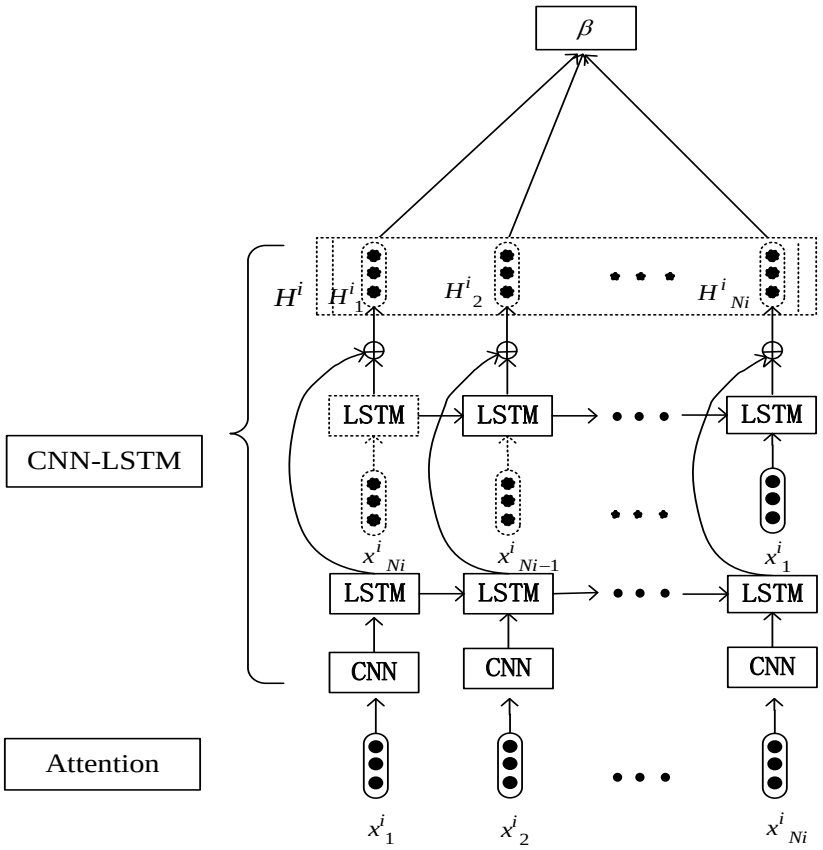

Fig. 5. Multi-model deep learning of attention-CNN-LSTM.

An attention model is a method that takes n arguments $y_1, y_2 \ldots\ldots\ldots y_n$ (in the precedent examples, the y_i would be the h_i), and a context c . It return a vector z which is supposed to be the summary of the y_i, focusing on information linked to the context c. More formally, it returns a weighted arithmetic mean of the y_i, and the weights are chosen according the relevance of each y_i given the context c.

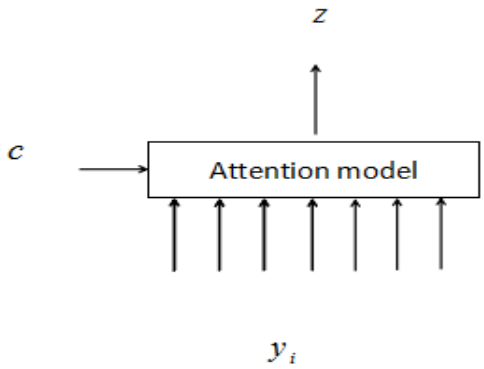

Fig. 6. Attention mechanism.

In deep learning filed, a convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks, most commonly applied to analyzing visual imagery. CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing. They are also known as shift invariant or space invariant artificial neural networks, based on their shared-weights architecture andtranslation invariance characteristics. A convolutional neural network is consists of many convolution layer and a pooling layer, as shown in Fig. 7.

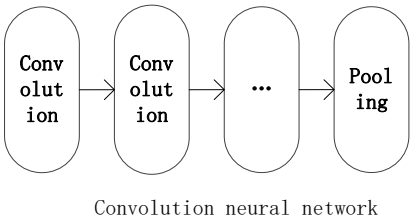

Fig. 7. Convolutional neural network.

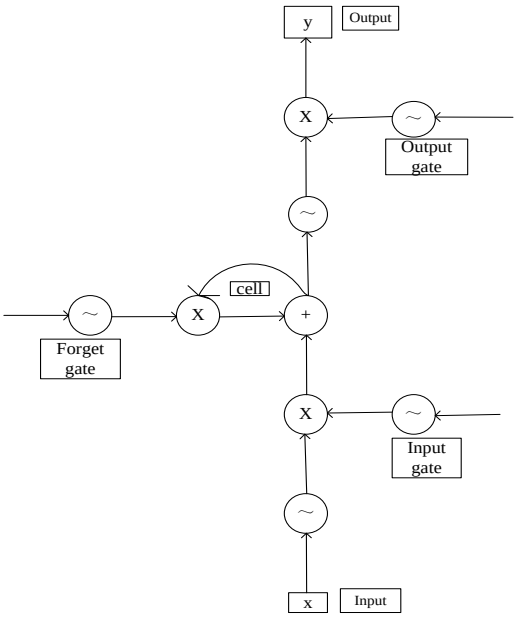

Fig. 8. LSTM model.

The Recurrent Neural Network (RNN) is a powerful family of connection models that captures temporal dynamics through cycles and learns context-sensitive information during the mapping of input and output

sequences. However, the range of context information that can be accessed is limited. Long and short-term memory (LSTM) networks is a good solution to these problems. It avoids long-term dependency issues by adding input gates, forget gates, and output gates to maintain timely updates. Long and short-term memory is an enhanced version of a component that is placed in a cyclic neural network. The basic structure of LSTM [25] is shown in Fig. 8.

The LSTM update formula at time t is as follows:

$$\mathrm{i}_t = \sigma(w_i h_{t-1} + u_i x_t + \mathrm{b}_i) \tag{1}$$

$$f_t = \sigma(w_f h_{t-1} + u_f x_t + \mathrm{b}_f) \tag{2}$$

$$\tilde{c_t} = \tanh(w_c h_{t-1} + u_C x_t + b_c) \tag{3}$$

$$c_t = f_t \Theta c_{t-1} + i_t \Theta \tilde{c}_t \tag{4}$$

$$o_t = \sigma(w_o h_{t-1} + u_o x_t + b_o) \tag{5}$$

$$h_t = o_t \Theta \tan \mathrm{h}(c_t) \tag{6}$$

where σ is a nonlinear sigmoid function, it is a point multiplication operation between two vectors. *Xt* is an input vector at time t and h_t is a hidden state vector that stores all useful information at time t. U_i, U_f, U_c, Uo denote the weight matrix of the different gates of the input X_t. W_i, W_f, W_c, W_o are the weight matrix of the hidden state h_t. b_i, b_f, b_c, b_o represent partial vectors.

On the basis of part 3.1, The two types of features extracted above (Malware Texture Fingerprint, Malware Activity Embedding in Vector Space) are input into Attention-CNN-LSTM for classification.

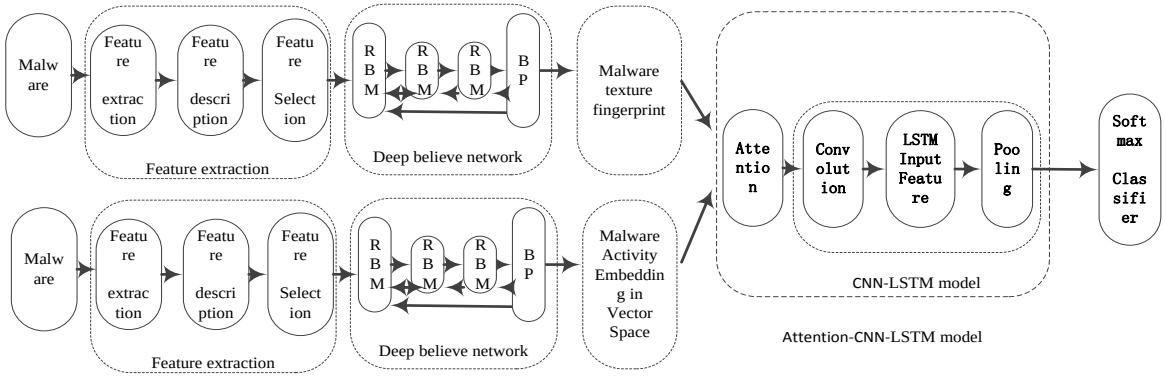

Fig. 9. Algorithm model.

4. Experiments

The experiment samples used for these experiments consisted of 5560 malicious and 123453 benign application. We select part of them in this paper. We can get the data sets from http://user.cs.uni-goettingen.de/˜darp/ drebin.

4.1. Experiments Environment

In order to identify the multi-model deep learning is applicable for Drebin malware analysis and detection. In experiment, we selected different data samples to determine variable parameters (feature number, number of

network layers, etc.), with 1550, 2620, 5825 experimental data, of which 60% of the sample data is 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 (7) is used to measure the performance of the model.

Table 1. Experiments Environment

Name	description	Value
OS	Operating System	Windows 7 64 bit platform
CPU	Central Processing Unit	Intel i7-7700@3.6HZ 3.6GHZ
GPU	Graphic Processing Unit	NVIDA GeForce GTX 1080
RAM	RamdomAccessMemory	32G
Hard disk	Hard disk	120G SSD+4T HDD
CUDA	Compute Unified Device Arcthitecture	7.5
CUDNN	NVIDIA CUDA® Deep Neural Network library	5.0
Python	Python	2.7.3
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

$$Q = (TP+TN)/(TP+TN+FP+FN) \tag{7}$$

In equation (7), *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.

4.2. Experiments Results

4.2.1. Effect of image texture on experimental results to accuracy

In order to verify the optimal dimensions of DBN under different texture feature dimensions, the experimental results are shown in Fig. 10 and Table 2.

Table 2. Effect of Image Texture on Experimental Results to Accuracy

	500	1000	2000	2500
1550	94	94.1	94.8	94.3
2620	94.08	94.9	95.1	94.6

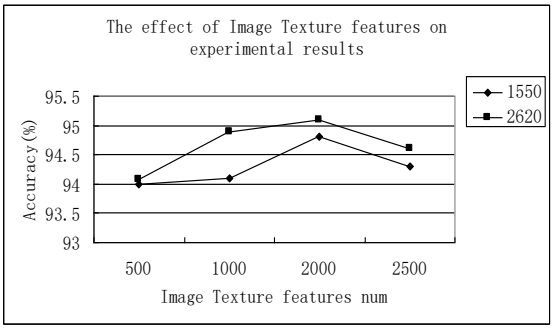

Fig. 10. Effect of image texture on experimental results.

4.2.2. The impact of DBN network layers on experimental results

In order to verify the accuracy of DBN classification under different network layers, combined with texture fingerprint features and malicious code activity vector space features, the following comparative experiments were carried out. The experimental results are shown in Table 3 and Fig. 11.

Table 3. The Impact of DBN Network Layers on Experimental Results

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

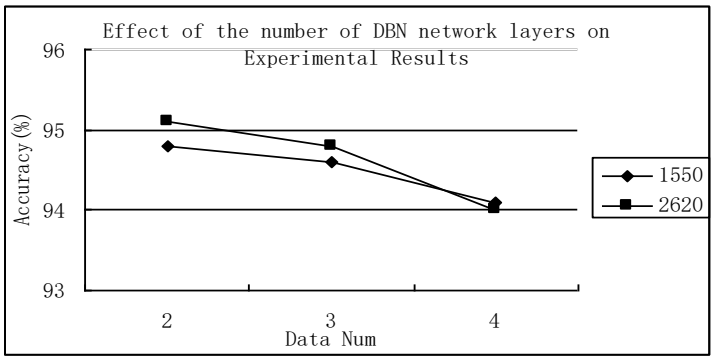

Fig. 11. The impact of DBN network layers on experimental results.

From the above experiments, we can find that when the texture fingerprint feature is 2000, the classification accuracy is the best, and when the DBN Network Layer is set to 2, the classification accuracy is the best.In the following experiments, the experimental model parameters are referred to Table 4.

Table 4. Parameters of the Model

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

4.2.3. The importance of texture image features to experimental results

To illustrate the importance of texture fingerprint features, the parameter models in Table 5 are applied.

Combined Malware Image Texture (MIT) and Malware Activity Embedding in Vector Space (MAEVS), Image Texture Median Filter (ITMF). The comparison was performed using only the malicious code activity vector space.The experimental results are shown in Fig. 12, Table 5.

Table 5 Effect of Texture Image 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

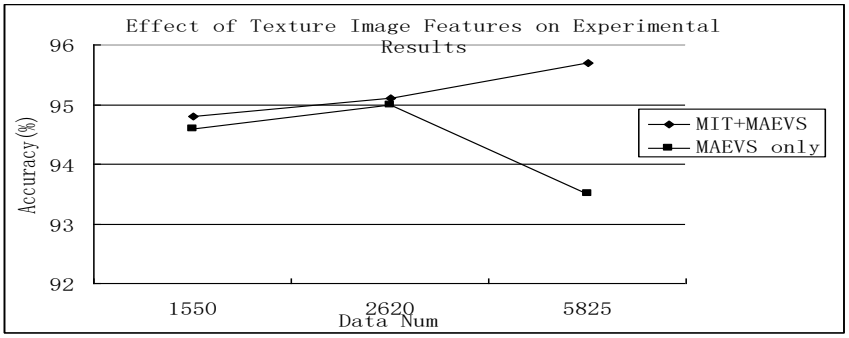

Fig. 12. Effect of texture image features on experimental results.

4.2.4. The algorithm of this paper is compared with other models

In order to verify the effectiveness of the Attention-CNN-LSTM Android malicious code classification algorithm proposed in this paper, combined with texture fingerprint feature MIT, texture fingerprint median filter feature ITMF, malicious code activity vector space MAEVS, the experimental results are as follows.

Table 6. Comparison with Shallow Machine Learning Model

Data num	Attention-CNN-LSTM (MIT+MAEVS)	DBN (MIT+ITMF+ MAEVS)	DBN (MIT+MA EVS)	SVM (MIT+MAEVS)	KNN (MIT+MAEVS)	ANN (MIT+MA EVS)	NDSS: Drebin [12]
1550	95.4	95	94.8	92.7	94.5	93.6	
2620	95.8	95.4	95.1	94.2	95	95	
5825	96.0	95.9	95.7	94.5	95.1	95.2	
6956	96.4	95.2	95.6	94.8	95.4	95.0	
average	95.9	95.375	95.3	94.05	95	94.7	94

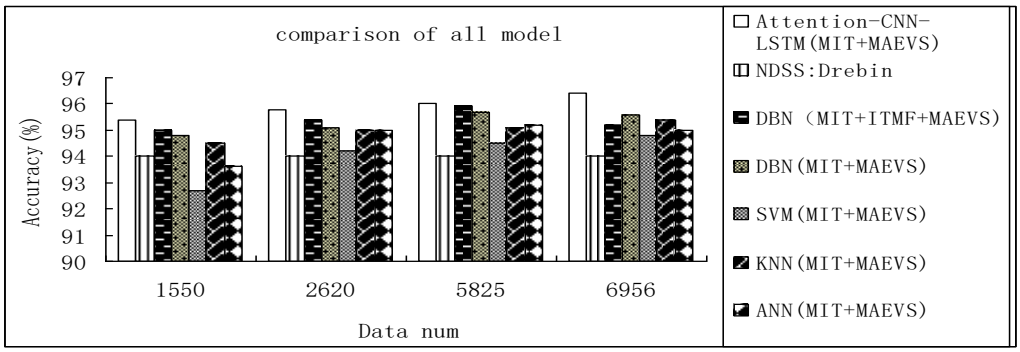

Fig. 13. Comparison with shallow machine learning model.

5. Conclusion

This paper proposes a malware detection algorithm based on multi-model deep learning of Attention-CNN-LSTM. Firstly, the Deep Belief Network is used to extract the malware texture fingerprint feature, Malware Activity Embedding in Vector Space, and then input these features into the Attention-CNN-LSTM for classification. The experiment results show that the proposed algorithm can detect malware effectively , and its average classification accuracy is significantly higher than single deep learning model DBN and shallow machine learning model SVM, KNN, ANN.

Acknowledgment

This papers is supported by the Ministry of Education of Humanities and Social Science project (17YJAZH043), the Natural Science Project of Hubei provience (2018CFB456), the Project of Cernet Next Generation Internet Technology Innovation Project (NGII20170420), the Research Innovation Project of Graduate Student in Xinjiang Uygur Autonomous Region (XJGRI2017007), the Natural Science Project of Xinjiang Uygur Autonomous Region (2015211C263), the Project of Teaching and Research in Hubei Polytechnic University (2016C08).

References

[1] Çelenli, N., Topgül, O., & İbrahim, H. (2016). Cloud-based malware analysis system for smart phones. *Signal Proceedings of Processing and Communication Application Conference, IEEE* (pp. 2057-2060).

[2] Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2018). Detecting Android malware using long Short-term memory (LSTM). *Journal of Intelligent & Fuzzy Systems, 34(3),* 1277-1288.

[3] Ham, H. S., & Choi, M. J. (2013). Analysis of Android malware detection performance using machine learning classifiers. *Proceedings of International Conference on ICT Convergence*, *IEEE* (pp. 490-495).

[4] Faruki, P., Ganmoor, V., & Laxmi, V. (2013). AndroSimilar: Robust statistical feature signature for Android malware detection. *Proceedings of International Conference on Security of Information and Networks, ACM* (pp. 152-159).

[5] Sahin, D. O., Kural, O. E., & Akleylek, S. (2018). New results on permission based static analysis for Android malware. *Proceedings of the International Symposium on Digital Forensic and Security* (pp. 1-4).

[6] Yang, H. Y., & Jin, X. U. (2018). An Android malware static detection model. *Journal of Jilin University,* 2018.

[7] Yan, L. K., & Yin, H. (2013). DroidScope: Seamlessly reconstructing the OS and Dalvik semantic views for dynamic Android malware analysis. *Proceedings of the 21st USENIX Conference on Security Symposium*, *USENIX Association* (p. 29).

[8] Petsas, T., Voyatzis, G., & Athanasopoulos, E. (2014). Rage against the virtual machine: Hindering dynamic analysis of Android malware. *Proceedings of European Workshop on System Security, ACM* (pp. 1-6).

[9] Sugunan, K., Kumar, T. G., & Dhanya, K. A. (2018). *Static and Dynamic Analysis for Android Malware Detection.*

[10] Onwuzurike, L., Almeida, M., & Mariconti, E. (2018). *A Family of Droids: Analyzing Behavioral Model Based Android Malware Detection via Static and Dynamic Analysis*.

[11] Shim, J., Lim, K., & Cho, S. J. (2018). *Static and Dynamic Analysis of Android Malware and Goodware Written with Unity Framework.*

[12] Zhang, M., Duan, Y., & Yin, H. (2014). *Semantics-Aware Android Malware Classification Using Weighted Contextual API Dependency Graphs*, *7(9),* 1105-1116.

[13] Hou, S., Saas, A., & Chen, L (2017). Deep4MalDroid: A deep learning framework for Android malware detection based on linux kernel system call graphs. *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence Workshops* (pp. 104-111).

[14] Atici, M. A., Sagiroglu, S., & Dogru, I. A. (2016). Android malware analysis approach based on control flow

graphs and machine learning algorithms. *Proceedings of International Symposium on Digital Forensic and Security, IEEE* (pp. 26-31).

[15] Wang, Z., & Wu, F. (2015). Android malware analytic method based on improved multi-level signature matching. *Proceedings of International Conference on Information Science and Technology, IEEE* (pp. 93-98).

[16] Oh, T., Jadhav, S., & Kim, Y. H. (2015). Android botnet categorization and family detection based on behavioural and signature data. *Proceedings of International Conference on Information and Communication Technology Convergence*, *IEEE* (pp. 647-652).

[17] Alam, S., Qu, Z., & Riley, R. (2016). *DroidNative: Semantic-Based Detection of Android Native Code Malware.*

[18] Zhang, X., & Jin, Z. (2017). A new semantics-based android malware detection. *Proceedings of IEEE International Conference on Computer and Communications, IEEE* (pp. 1412-1416).

[19] Cade, C. (2015). Understanding heuristicbased scanning vs. Sandboxing. *Database & Network Journal.*

[20] Ali, M., Ali, H., & Anwar, Z. (2012). Enhancing stealthiness & efficiency of android trojans and defense possibilities (EnSEAD) - Android's malware attack, stealthiness and defense: An improvement. *Frontiers of Information Technology, IEEE* (pp. 148-153).

[21] Jadhav, S., Dutia, S., & Calangutkar, K. (2015). Cloud-based Android botnet malware detection system. *Proceedings of International Conference on Advanced Communication Technology, IEEE* (pp. 347-352).

[22] Luo, S., Tian, S., Yu, L., Yu, J., & Sun, H. (2018). Android malicious code classification using deep belief network. *KSII Transactions on Internet and Information Systems, 12(1),* 454-475.

[23] Luo, S., & Tian, S. (2017). Research strategy of classify malicious code into families on the method of deep belief networks. *Journal of Chinese Computer Systems, 38(11),* 2465-2470.

[24] Geoffery, E., & Hinton, S. R. (2006). Reducing the dimensionality of data with neural networks. *Science, 313(5786),* 504-7.

[25] Greff, K., Srivastava, R. K., & Koutnik, J. (2017). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks & Learning Systems, 28(10),* 2222-2232.

Luo Shi-qi was born in Hubei of China in 1993. He received the master dgree in teacher at School of Computer, Hubei Polytechnic University. He received the master degree in Xinjiang University. His main research interests include information security.

Liu Zhi-yuan was born in Hubei of China in 1972. He received the Ph.D dgree, and is a professor in School of Computer, Hubei Polytechnic University. His main research interests include information security.

Ni Bo was born in Hubei of China in 1982. He received the Ph.D dgree, and is an associate professor in School of Computer, Hubei Polytechnic University. He received the Ph.D degree in Wuhan University. His main research interests include computer vision.

Wang Huan-huan was born in Henan of China in 1995. She is a postgraduate student in the School of Software, Xinjiang University. Her main research interests include information security.

Sun Hua was born in Xinjiang of China in 1974. She received the Ph.D degree, and is an associate professor in School of Software, Xinjiang University. She received the Ph.D degree in East China University of Science and Technology. Her research interests include information security.

Yuan Yong was born in Hubei of China in 1983. He is a lecturer in School of Computer, Hubei Polytechnic University. His main research interests include human computer.