UTM Computing Proceedings
Innovations in Computing Technology and Applications
Volume: 6 | Year: 2024 | ISBN No. : 2976-2278

Android Malware Classification Using Deep Learning Methods On Static Features

Fitri Zulkarnain Bin Zaharen, Aida Ali, and Howida Abu-baker
Faculty of Computing,
Universiti Teknologi Malaysia,
81310 Johor Bahru, Malaysia
fzzaharen@graduate.utm.my, aida@utm.my, howida10@gmail.com

***Abstract*— The information security community has been attempting to increase the effectiveness and efficiency of malware classification. Many sectors could benefit from the development of an anti-malware system that can combat unidentified malware. Deep learning (DL) models enable the development of intelligent anti-malware solutions, which can be proposed. It might be able to mathematically generalize the identification of recently released malware by using such models to establish a link between a particular malware family x and the malware family y that it is a member of f: x 7 y. To categorize each malware family, CNN and M-CNN DL models are trained. In order to achieve this, Malimg dataset is used which made up of malware images that were processed from malware binaries.**

Keywords- artificial intelligence; artificial neural networks; machine learning; malware classification; deep learning

I. INTRODUCTION

As long as it is regarded as any programmable device, malware refers to any kind of malicious software that is designed to harm a system or even a network. The reason for this is that malware sometimes contains harmful executable code that is capable of eradicating the system's default or regular services and functions [1]. Cybercriminals typically utilize malware to further their own individualized or collective goals. Important information will be taken from victims for nefarious purposes, such as gathering information for identity theft by luring victims into providing it, gathering credit card or other financial data, launching denial-of-service attacks against other networks powered by controlled computers, and many other things. These crucial details are typically presented in the form of financial information, emails, passwords and even healthcare records.

Researchers and security professionals were compelled by the malware's growth to create new classification schemes to separate these distinct varieties of malware, particularly in terms of features. For the objective of classifying malware, numerous classification techniques, including deep learning and statistical analysis, have been developed. Deep learning techniques use several kinds of multilayer networks that learn representations at various degrees of abstraction [2].

This paper will concentrate on the investigation of deep learning-based approaches for classifying Android malware. It will be possible to undertake thorough analysis of the present systems for categorizing Android malware by doing research based on previously gathered material. This study will be able to supplement earlier research by addressing some knowledge gaps and introducing some fresh concerns to the field.

II. MALWARE FAMILY

Based on McAfee's report, the malware application of Android has increased to 22 million in 2017. The most dominant mobile Operating System is Android as its market leads by capturing 86% of the total market [5]. Symantec also reported that every 1 of 5 Android applications can be considered as malware. It is bold to assume that Android application security has to play its crucial role especially for the researchers in order to defend their users from malicious malware developers.

A. AnserverBot

AnserverBot refers to an Android Trojan that has been considered as one of the most complicated Android malware. Usually it will piggyback on legitimate applications which are currently being used in popular third-party Android especially in China. It is worth paying attention to this Trojan in terms of several aspects as it is developed with some complicated techniques that are significant in evading analysis or even detection. Furthermore, this malware program is active in fetching commands from encrypted blog posts as it has built-in bot functionality. Due to the matter of this fact, the bar for reverse engineering analysis has been raised. In Android malware history, AnserverBot has been known as the prime malware that is able to apply the mentioned technique efficiently integrated into one real-world instance.

B. DroidSheep

DroidSheep is known as a tool used for listening to network traffic by threat actors. It allows them to gain specific access into online accounts of numerous popular websites on the internet. By running this malware, threat actors will be able to impersonate victims' accounts thus accessing sites without using a secure connection. It also allows threat actors to sniff wireless network traffic which later benefits them in stealing authentication tokens. As a result, the threat actors once again will be able to impersonate the victim. However, the good news is DroidSheep is unable to infiltrate current popular sites such as Yahoo, Facebook and Google as long as it supports HTTPS connections.

C. DroidKungFu

DroidKungFu refers to a malware that affects Android Operating System which primarily targets the users in China. This malware capables of evading current anti-virus software thus rooting the vulnerable Android phones without any problems [4]. It is noteworthy because DroidKungFu encrypts two known roots which is Udev Exploit and RageAgainsTheCage exploit in order to break the security mechanisms. It will decrypt these two exploits before executing them, which refers to launching an attack when it runs. Repackaged applications have the attachment of DroidKungfu inside it that is available in numerous alternative application markets. As a result, a new service and a new receiver will be added into the infected application. Later on, a notification regarding the system state of booting will be sent to the receiver so that the service can be launched without any user interactions as the system finishes booting.

III. DEEP LEARNING

The multilayer networks that make up the deep learning model are connected to various levels of abstraction when learning data representations. [2]. Although the invention of neural networks served as the foundation for the rise of deep learning, several research have produced conflicting results regarding how the two are related. Machine learning that is based on neural networks is called deep learning. However, there is a claim that deep learning is the sort of neural network technology that is most frequently used.

Neural networks replicate the biological neurons that communicate with the outside world [6]. Walter Pitts is a logician, while Warren McCulloch is a neurophysiologist. Both of them were inspired to propose the conceptual "M-P" neuron model by the structure of biological neurons [7]. In a biological neural network, the output values "0" or "1" denote the two states of neuronal inhibition and activation. A neural network is created by connecting several neurons together in a layered structure. Neural networks are regarded as mathematical models with many characteristics for use in macroscopic mathematical computation. Neural network-based machine learning falls into two primary groups. The first one is based on the perceptron model [7]. Using a multilayer perceptron, one can train a network to get the desired results while taking use of the back propagation algorithm's mistake [9]. The Boltzmann machine serves as the foundation for the second [10]. A neural network at random that generates results according to probability distribution.

A. Convolutional Neural Network

It has learnable parameters within the hidden layers of neurons, such as inputs, performs a dot product, and exhibits non-linearity, which makes it similar to feedforward neural networks. These parameters will be sent to the neurons. The entire network will represent the mapping between the unprocessed image pixels (x R m) and the class grades (y, f: x 7 y). Based on the size of the layer inputs and outputs, the usage of ReLU, and the use of the traditional Softmax function as a network classifier, the design of this method is modified [11].

B. Gated Recurrent Unit

The vanishing gradient problem that occurs with a standard recurrent neural network is the goal of the gated recurrent neural network architecture [13]. Long short term memory (LSTM) is a version of this since both designs are equally good at producing results. To address the disappearing gradient issue, it used update and reset gates. The information that should be sent to the output will be determined by the two vectors. As an illustration, the network architecture is built using a mix of a support vector machine (SVM) and a form of a recurrent neural network (RNN) as well as a gated recurrent unit (GRU) [12]. A deeper understanding of the deep learning approach will be gained by studying the behavior of this model.

C. Convolutional Neural Network

It is a neural network that resembles CNN and has a hidden layer of perceptrons. However, the VGG-16 [14] is the foundation of the model architecture. Every network starts with a stack of convolutional layers, followed by three fully connected layers with the same configuration, and then the soft-max layer, which serves as the top-level layer. Every hidden layer also has rectification non-linearity (ReLU) installed.

TABLE II. COMPARISON BETWEEN METHODS

Index	*Reference*	*Features*	*Method*	*Limitation*
1	[17]	Battery	Recurrent Neural Network (RNN)	Difficult to be trained
2	[18]	Grayscale Image	Convolutional Neural Network (CNN)	Lack of hidden layers
3	[15]	Grayscale Image	K-Nearest Neighbor (KNN)	Memory intensive
4	[19]	API	C4.5	Information entropy
5	[16]	Grayscale Image	M-Convolutional Neural Network (M-CNN)	Need a lot of data for training

IV RESEARCH WORKFLOW

To compare the performance of CNN and M-CNN in classifying Android malware, several phases and activities are prepared to ensure the study is conducted systematically. The research framework is divided into five phases which are firstly about the research planning and initial study. The second phase is data preprocessing followed by feature selection in the third phase. The next phase is dedicated for the model development before pursuing for the testing and evaluation on the last phase. The performance measures of the classification technique is

used in this research as a measurement to compare both models in classifying Android malware. The general research framework is shown as in Fig.1.

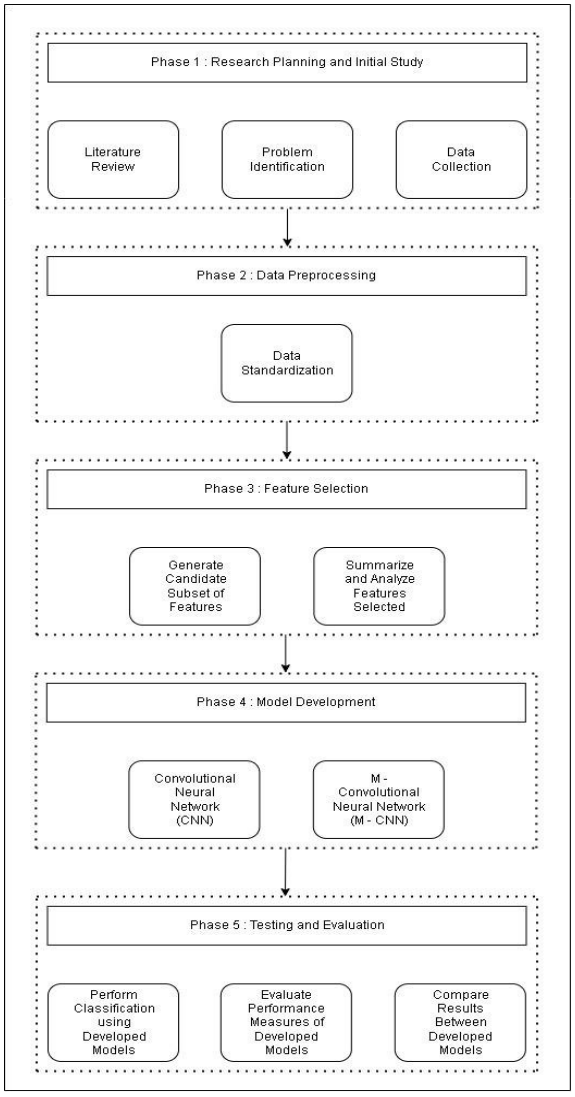

Figure 9. Research framework

A. *Research Planning and Initial Study*

The information acquired from the literature allowed for a better understanding of the trends and tendencies in the study field, which is essential to identifying the difficulties and problems. As a result, research objectives and scopes could then be determined. Setting the course of the research is the goal of research objectives. There are three research aims in this study, as was described in chapter one. Exploring the research field to ensure that it is neither too large nor too tiny is known as the research scope. The goals and parameters of this study are discussed in chapter one of this documentation.

B. *Data Preprocessing*

To accomplish this research, a dataset is required in order to evaluate the model in classifying features. The dataset will be used which consists of malware and non-malware samples. The binary form of the samples will be processed to grayscale images. It is created through the reading of malware binaries into an 8-bit unsigned integer composing a matrix of $M \epsilon Rm \times n$. The matrix will be visualized in a grayscale image form which has a range of values [0,255]. From the grayscale image, Black color can be interpreted as 0 while white color as 1. Data preprocessing is significant in order to obtain maximum efficiency in malware analysis. It is because the raw data will be unable to support machine learning [20].

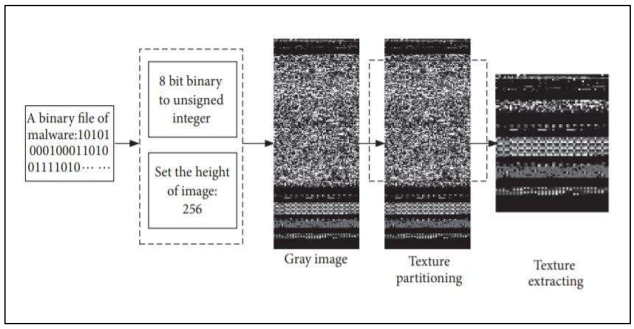

Figure 10. Conversion of binary file to grayscale image

The malware images were rescaled to a 2-dimensional matrix and then compressed into a n x n-size array, yielding a 1 x 1024-size array [21]. The indexed malware family name was labeled to each of the feature arrays that corresponded to it. The features were then standardized using Eq.1.

$$z = \frac{x - \mu}{\sigma} \quad (1)$$

Where μ refers to the mean value, σ is the standard deviation and is its feature to be standardized. Although the dataset is made up entirely of images, and standardization may not be appropriate for such data, keep in mind that the images are taken from malware binary files. As a result, the features aren't actually images to be started with.

The feature selection is a well-known way in order to reduce the dimension of the dataset. By removing the unnecessary and redundant features, the number of features can be limited which is giving a positive impact towards the dataset. Hence, the performance of generalization and operational efficiency can be improved by selecting an optimal subset of features. In general, the steps of feature selection can be divided into generation, evaluation, stopping criterion and validation [22].

Features can be analyzed and summarized according to three common categories which are static features, dynamic features and hybrid features. Most of the time, it depends on whether they are acquired by running an Android application or not [23]. By doing selection on the dataset, all of the selected features are in the form of grayscale images. It can be classified into the category of static features. In terms of static features, it refers to features that can be generated through the process of code analysis [24]. Usually, researchers will select either permissions, API call, opcode sequence or function call graphs

as features. The permission features are the best single predictor of the application malignity [25]. However, the combination of visualization-based analysis and deep learning have provided a great impact in the research field related to privacy and security lately [26]. Most of the proposed solutions are able to achieve high accuracy against windows malware classification [17]. As a result, this research will validate the classification of Android malware using grayscale images.

D. Model Development

Convolutional Neural Network (CNN) consists of hidden layers of neurons with specific parameters that can be learned such as inputs, dot product and non-linearity. The whole network can be expressed as mapping between raw image pixels x ϵ Rm and class scores y, f : x 7→ y. The modification on the architecture of this method is made based on the size of layer inputs and outputs, the use of ReLU with conventional Softmax function as network classifier [11].

E. Testing and Evaluation

The performance metrics can be obtained by referring to typical binary classification which refers to the results of a prediction that can be divided into four types which are false positive, true positive, false negative, and true negative [27]. False negative can be an indicator of the incorrect prediction number in a recurrent malware class while true negative indicates the correct prediction number in a recurrent malware class. Next, false positive will be referred to as the incorrect prediction number in a primary malware class while true positive refers to the correct prediction number in a primary malware class. The performance of the model was evaluated by using some important variables such as accuracy, precision and recall.

The most important performance metrics are precision and recall. However, both of them only provide a partial evaluation. It is because precision will represent the ratio of all samples correctly predicted to be positive among all samples predicted to be positive while recall will represent the ratio of all positive samples correctly predicted among all positive samples.

$$\text{Precision } (P) = \frac{TP}{TP + FP} \qquad (2)$$

$$\text{Recall } (R) = \frac{TP}{TP + FN} \qquad (3)$$

V. EXPERIMENT DESIGN

The experimental design starts with data preprocessing where the data were rescaled, compressed before generating it into desired output for analysis. Next step is generating the dataset from Google Drive directory followed by standardizing target size and batch size of the dataset. Quick analysis was made in order to evaluate the dataset according to its classes. Furthermore, the generated batches were split between training and testing. Lastly, the testing and evaluation phase is where the performance measures of both models are compared in terms of classifying malware. All methodology steps below are done using Python programming language in the Google Colaboratory environment. The experiment workflow is shown as in Fig.3.

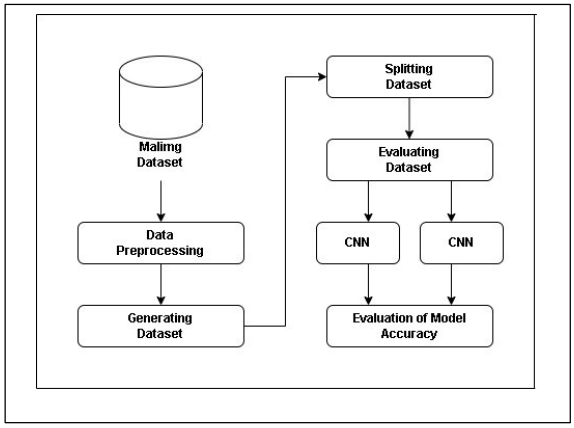

Figure 11. Experiment workflow

All experiments in this study were conducted on a laptop computer with Intel® Core™ i5-7300HQ CPU @ 2.50GHz, 4GB of DDR4 RAM and NVIDIA GeForce GTX 1050. Table II shows the hyper-parameters used by the models in the conducted experiments.

The Malimg dataset was generated by using the ImageDateGenerator () from Keras that allows to perform image augmentation. It was done by producing batches of normalized tensor image data from the respective malware families. All images were resized to the specified size of 224 x 224 x 3 and the batch size that was selected is 1000. The list of generated malware is shown in Fig.4.

```
[ ] batches.class_indices

{'Adialer.C': 0,
 'Agent.FYI': 1,
 'Allaple.A': 2,
 'Allaple.L': 3,
 'Alueron.gen!J': 4,
 'Autorun.K': 5,
 'C2LOP.P': 6,
 'C2LOP.gen!g': 7,
 'Dialplatform.B': 8,
 'Dontovo.A': 9,
 'Fakerean': 10,
 'Instantaccess': 11,
 'Lolyda.AA1': 12,
 'Lolyda.AA2': 13,
 'Lolyda.AA3': 14,
 'Lolyda.AT': 15,
 'Malex.gen!J': 16,
 'Obfuscator.AD': 17,
 'Rbot!gen': 18,
 'Skintrim.N': 19,
 'Swizzor.gen!E': 20,
 'Swizzor.gen!I': 21,
 'VB.AT': 22,
 'Wintrim.BX': 23,
 'Yuner.A': 24}
```

Figure 12. List of malware classes

As the 25 different classes of malware are recognized, the dataset is evaluated using the Matplotlib from Python. The classes of malware generated through the batches of grayscale images. The percentage of malware classes obtained through the sum of labels over the number of label's dimensions in array. The horizontal axis represents the independent variable which is the malware class. Meanwhile, the vertical axis

represented the percentage of malware class. The bar graph formed from the percentage of malware classes is shown in Fig.5.

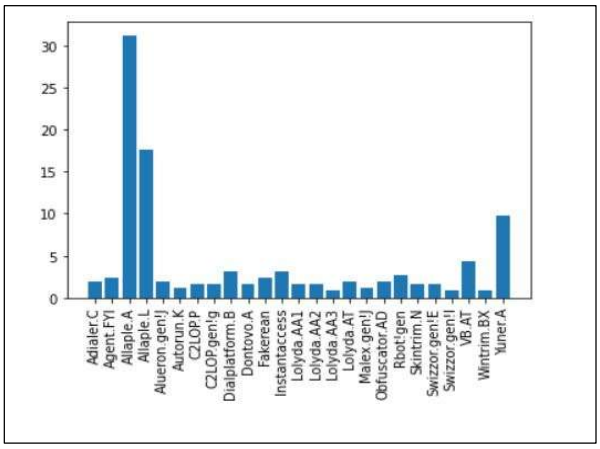

Figure 13. Percentage of malware classes

According to the Fig.5, the dataset is in an unbalanced state. It is because there are more than 30% images belonging to class 2 which refers to Allaple.A and 17% images belonging to class 3 which refers to Allaple.L. Table II shows the hyper-parameters used by both models in the conducted experiments.

TABLE II. HYPER-PARAMETERS USED IN BOTH MODELS

Hyper-parameters	***CNN***	***M-CNN***
Batch Size	1000	1000
No. of Hidden Layers	2	5
Epochs	10	10

The Malimg dataset is split into training and testing datasets as the initial stage before training the model. All of the models underwent 10-epoch training on 6537 malware family variants (6400 mod 256 = 0), which made up about ≈70% of the preprocessed dataset. However, only ≈30% of the preprocessed dataset which refers to 2560 malware family types (2560 mod 256 = 0) was used to test the models over the course of 10 epochs. Using the classification_report() function of sklearn.metrics, the classification measures accuracy, precision, and recall were all calculated [28]. The experiment findings for the models that were given are compiled in Table III.

TABLE III. PARAMETERS USED IN BOTH MODELS

Variables	***CNN***	***M-CNN***
Accuracy	43.33%	50.00%
Precision	0.24	0.27
Recall	0.43	0.50

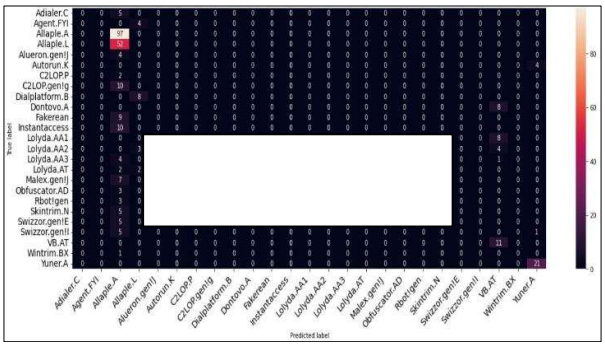

Figure 14. Confusion matrix of CNN

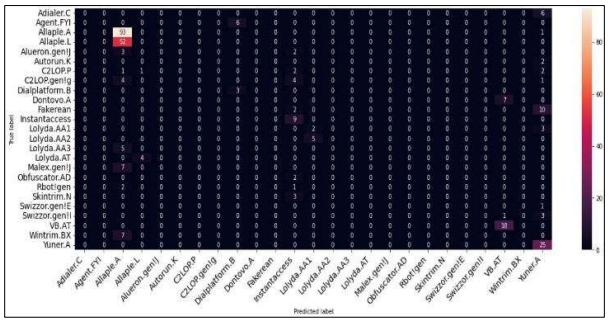

Figure 15. Confusion matrix of M-CNN

Table III shows the testing performance of CNN and M-CNN models in multinomial classification on malware classes. M-CNN had a test accuracy of ≈50.00%, a precision of 0.27 and a recall of 0.50. The table also shows the testing performance of CNN having a test accuracy of ≈43.33%, a precision of 0.24 and a recall of 0.43. All the models scored higher for Allaple.A which is the malware with the greatest number of variants, as can be seen in the confusion matrices. In this instance, it might be because relative populations of each malware family were not taken into account when data partitioning was used to separate the dataset into training and testing data.

However the Allaple.A is frequently mistaken for the Allaple.L. In actuality, they both had worm-type backgrounds and had the same trait. As a result, there might have been similarities in their code. The Fakerean, on the other hand, has consistently been mislabeled as a Yuner.A. It's because the dataset only contains a small number of examples of Fakerean. Additionally, these malwares share a Trojan-type background, which is where they both originated.

VI. DISCUSSION

It is noticeable that the M-CNN model stands out more compared to CNN model presented in this study. This finding comes as no surprise as the M-CNN is constructed with relatively the most sophisticated architecture design among the presented model. Particularly, it is because of its 5-layer design. In terms of neural networks, the number of layers residing is directly proportional to the complexity of a function it can portray [29]. In a specific way, the performance of a neural network is directly proportional to the number of its hidden layers. This logic describes the hypothesis which is the less number of hidden layers that a neural network possesses, the

less its performance is. As a result, the findings in this study supported the literature explanation as the CNN came second (≈43.33% test accuracy) with a 2-layer design.

According to the results in Table III, the test accuracy of≈50.00% clearly shows that the M-CNN model has the highest predictive performance compared to the CNN. As a matter of fact, M-CNN has a more complex design compared to its rival. Firstly, it consists of 5-layer design that lets it perform complex mappings between features and labels. Secondly, it has the capabilities in learning from data of sequential nature especially for which an image data belongs. Thirdly, all hidden layers are equipped with the rectification non-linearity as its model architecture is based on VGG-16 [14]. As a result, it had longer computing time compared to CNN as it had more non-linearities introduced in its design. The CNN model might be able to improve its predictive performance by adding more hidden layers and better non-linearities to the architectural design. Thus, the complexity will be able to support the model to be on par with the M-CNN model.

VII. CONCLUSION

The research is an exercise to explore the model performance of CNN and M-CNN in classifying malware from 25 different classes in the Malimg dataset. Further research is needed to come out with valid and promising models and results. Based on the results achieved from Table III, it shows that M-CNN model had the higher predictive accuracy compared to CNN model, having a test accuracy of ≈50.00%. There are several that can be made in order to improve the outcomes in the future. Since the accuracy of M-CNN outperforms CNN, the architecture design of CNN may provide greater insights on malware classification by adding more hidden layers, improving non-linearities and employing an optimal dropout. Such ideas in improving the architecture design of the models may give an idea on which architecture will serve best in the engineering of adaptive anti-malware systems [12].

ACKNOWLEDGMENT

We would like to express our sincere gratitude to Faculty of Computing and Universiti Teknologi Malaysia (UTM) for their invaluable support and resources, which played a crucial role in the successful research.

REFERENCES

[1] M. Sikorski, and A. Honig, Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software. San Francisco, CA, USA: No Starch Press, 2012.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436-444, May 2015.

[3] Lakshmanan Nataraj, S Karthikeyan, Gregoire Jacob, and BS Manjunath. 2011. Malware images: visualization and automatic classification. In Proceedings of the 8th international symposium on visualization for cyber security. ACM, 4.

[4] Isohara, T.; Takemori, K.; Kubota, A. Kernel-based behavior analysis for android malware detection.

[5] Gartner Says Worldwide Sales of Smartphones Recorded First, Ever Decline During the Fourth Quarter of 2017. Available online:https://www.gartner.com/newsroom/id/3859963 (accessed on 1 April 2018).

[6] T. Kohonen, "An introduction to neural computing," Neural Netw., vol. 1, no. 1, pp. 3-16, Jan. 1988.

[7] W. S. Mcculloch, and W. H. Pitts, "A logical calculus of the ideas immanent in nervous activity," Bull. of Math. Biol., vol. 5, pp. 115-133, Dec. 1943.

[8] D. O. Hebb, The Organization of Behavior-A Neuropsychological Theory. Hoboken, NJ, USA: John Willey and Sons Inc., 1949.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning Internal Representations By Error Propagation. Cambridge, MA, USA: MIT Press, 1988.

[10] N. L. Roux, and Y. Bengio, "Representational power of restricted boltzmann machines and deep belief networks," Neural Comput., vol. 20, no. 6, pp. 1631-1649, Jun. 2008.

[11] Yichuan Tang. 2013. Deep learning using linear support vector machines. arXiv preprint arXiv:1306.0239 (2013).

[12] Abien Fred Agarap. 2017. A Neural Network Architecture Combining Gated Recurrent Unit (GRU) and Support Vector Machine (SVM) for Intrusion Detection in Network Traffic Data. arXiv preprint arXiv:1709.03082 (2017).

[13] J. Y. Kim, S. J. Bu, and S. B. Cho, "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders," Inf. Sci. (Ny)., vol. 460–461, pp. 83–102, 2018

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556,2014.

[15] J. Yan, Y. Qi, and Q. Rao, "Detecting malware with an ensemble method based on deep neural network," Secur. and Commun. Netw., vol. 2018, pp. 7247095(16pp), Mar. 2018.

[16] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang and F. Iqbal, "Malware Classification with Deep Convolutional Neural Networks,"2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2018, pp. 1-5, doi: 10.1109/NTMS.2018.8328749.

[17] Vinayakumar, R.; Soman, K.; Poornachandran, P.; Sachin Kumar, S. Detecting Android malware using long short-term memory (LSTM). J. Intell. Fuzzy Syst. 2018, 34, 1277–1288.

[18] Ahmed Lekssays, Bouchaib Falah, Sameer Abufardeh. A Novel Approach for Android Malware Detection and Classification using Convolutional Neural Networks. ICSOFT 2020: 606-614.

[19] Z. Yuan, Y. Lu, Z. Wang, and Y. Xue, "Droid-sec: deep learning in Android malware detection," in Proc. 2014 ACM Conf. SIGCOMM, Chicago, Illinois, USA, 2014, pp.371-372.

[20] K. Kobkul, C. Gareth, and M. Phayung, "A hybrid system based on a multi-agent system in the data preprocessing stage," Int. J. of Comput. Sci. and Inf. Secur., vol. 7, no. 2, pp. 199-202, Feb. 2010.

[21] Felan Carlo C. Garcia and Felix P. Muga II. 2016. Random Forest for Malware Classification.

[22] M. Dash, and H. Liu, "Feature selection for classification," Intell. Data Anal., vol. 1, pp. 131-156, Mar. 1997.

[23] A. Qamar, A. Karim, and V. Chang, "Mobile malware attacks: review, taxonomy & future directions," Future Gener. Comput. Syst., vol. 97, pp. 887-909, Aug. 2019.

[24] B. Amro, "Malware detection techniques for mobile devices," Int. J. of Mobile Netw. Commun. & Telematics, vol. 7, no. 4/5/6, pp. 1-10, Dec. 2017.

[25] S. Hahn, M. Protsenko, and T. Muller, "Comparative evaluation of machine learning-based malware detection on Android," in Lecture Notes in Informatics Sicherheit 2016-Sicherheit, Schutz und Zuverlässigkeit. M. Meier, D. Reinhardt, and S. Wendzel, Eds. 2016, pp. 77-88.

[26] Kasongo, S.M.; Sun, Y. A deep learning method with wrapper based feature extraction for wireless intrusion detection systems. Computer Security 2020, 92, 10172.

[27] E. P. Costa, A. C. Lorena, A. C. P. L. F. Carvalho, and A. A. Freitas, "A review of performance evaluation measures for hierarchical classifiers," presented at 22nd Nat. Conf. Artif. Intell., Vancouver, British Columbia, Canada, Jul. 22-26, 2007.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12 (2011), 2825–2830.

[29] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Deep Learning. MIT Press. http://www.deeplearningbook.org.