Deep Learning Algorithms for Android Malware Detection

Ouda Adomuha[1], Alalmai M. Abdulhadi[1], Alqahtani S. Ibrahim[1],
Ferenczi Tamas[1], Garza I. Jose[1]
Vahid Emamian[2], Senior IEEE Member
[1]Computer Science Department
[2]Electrical Engineering Department
St. Mary's University, San Antonio TX
aouda@stmarytx.edu

Abstract

Malware on android devices were discussed as a growing threat to security. Deep learning was introduced as a more effective approach to dealing with this problem. Static, dynamic and hybrid deep learning approaches in the analysis of malware were discussed along with several algorithms known till date. Several literatures were reviewed and analyzed to inform the efficiency and accuracy of various algorithms. It was quickly discovered however that based on different factors; implementation could vary. Some challenges were identified and concluded that though this approach to malware detection is more effective than traditional malware detection software, there is more work to be done in order to maximize its potential.

1.0 Introduction

Android, an open source software designed for mobile devices such as smartphones and tablets with primarily touchscreen as input interface has grown exponentially in the last decade. This growth has been a catalyst for the increased rate of malware attacks on these types of systems. Android malware is a common problem in today's phones and apps. It is getting harder to monitor and find all malware with human effort. Since traditional antivirus software have been deficient at tackling the dynamic nature of attacks on them, deep learning was introduced and discovered to be a better approach in dealing with this challenge. Deep learning, a form of machine learning has been discovered to make this task easier. It conducts a profound classification and improves its accuracy because deep learning identifies more features than conventional machine learning methods by passing through many levels of feature extraction. This enables deep learning models to acquire a new pattern of malware after the basic training phase. There are security and accuracy problems that arise with deep learning. The Google app store uses a malicious application detection system called Bouncer. It is said to be effective but there have been errors with it that let malicious app on the to store. Other stores like the Android Play-Store uses data from users like rating and comments, but it takes time to detect the malicious software which can do harm in the meantime. The bad guys try to hide their malware by using different methods like code obfuscation technique, encryption, including permissions

which are not needed by the application, requesting for unwanted hardware, download or update attack in which a benign application updates itself or another application now with malicious payload. Trying to manually check apps to see if it is malicious or not and relying on bad software detection has led to the need to machine learning. There 3 types of analysis static dynamic, hybrid. Static analysis is done without running an application. Some examples of static features include, permissions, API calls which can be extracted from the AndroidManifest.xml file. Dynamic analysis deals with features that were extracted from the application while running, including network traffic, battery usage, IP address. The third type of analysis is hybrid analysis which combines the features from static and dynamic techniques.

1.1 Android Application Components

Android applications are made in Java and C++. Every application is managed by XML descriptor document called AndroidManifest.xml. The Android Manifest document contains about the applications made for android. These applications comprised of four parts: Activity, Service, Content Provider, and Broadcast Receiver. The way android works is that it has four layers. They manage the system from hardware sensors to user apps. Each layer does something specific for apps and different functions. The first layer is the Linux Kernel. It helps the OS services and manages the hardware's functions like memory, power, drivers, network stack, security settings, shared libraries and hardware abstraction. The second layer is the native library. It has native libraries which help manage data processing. This layer has open source libraries, such as surface manager, media framework, SQLite, Webkit, OpenGL—ES, FreeType, and SSL. These libraries do things like composing windows on the screen, processing input and output of video and audio data, database operations, supporting web browsers, high performance 2D and 3D graphics, fonts support, services of SSL and TLS protocols. The third layer is the Application Framework. This layer includes the Android APIs. This layer works with running apps and helps manages the basic functions on the device. The programs in this layer are activity manager, content provider, telephony manager, location manager, and resources manager. Each of these managers do a specific task and do not interfere with each other. The fourth layer is the Application Layer. This layer is on the top of the stack. The application layer helps in making calls, managing contacts, sending messages, and browsing web. In this layer there are a set of core applications, such as email client, calendar, browser, maps, contacts, SMS program, gallery.

1.2 Android Malware Detection Techniques

Mobile phones are a sensor-based event system, which permits malware to respond to approaching SMS, position changes and so forth, increasing the sophistication of automated malware-analysis techniques. Apps can use services, activities, and integrate varied programming languages in one application.

1.2.1 Static Analysis

This type of analysis scans an app without running it. It looks for obvious and distinct signs of malware, so it falls short to recognize the variation or unidentified malware. It also looks for permissions an app wants to determine if it is malware. The drawbacks of static analysis are the

missing of real execution paths and suitable execution conditions. Additionally, there exist problems in the occurrence of code obfuscation and dynamic code loading.

1.2.2 Dynamic Analysis

Dynamic analysis technique includes the execution of the application on either a virtual machine or a physical device. While examination, the behavior of the application is watched and can be dissected. The principle objective of the analysis is to achieve high code inclusion since every feasible event ought to be activated to watch any possible malicious behavior.

1.2.3 Hybrid Analysis

The hybrid analysis technique includes consolidating static and dynamic features gathered from examining the application and drawing data while the application is running.

2.0 Related Work

Several works have been done to understand how deep learning can be used to address the problem of malware infestation in android devices. Among them are a few listed below.
Chen, T. et al used Deep Neural Network (DNN) in a multimodal approach to malware detection and classification in smart phone devices to solving the problem of malware detection that involves the incorporation of different features.
Markel, Z. and Bilzor, M in their research, conducted a study to determine the efficiency and accuracy of using DNN to detect malware in android devices where DNN algorithm-based approach achieved an accuracy level of 95.31%. Martin, A. et al also conducted a study on the use of deep neural networks architecture to identify and classify malware classification
Hsien-De Huang, T. and Kao, H. Y in 2018 sought to develop a convolutional neural network system that could learn to detect malware in android devices without having to first extract features.
Baskaran B. and Ralescu A. identified the update attack as the most difficult intrusion technique and therefore used a review of existing literature to determine the best approach to address this challenge. Several types of malware were discussed and ways to defend against them. B. Baskaran and A. Ralescu talks about different ways and methods to scan and monitor apps. There three types of analysis are static dynamic, hybrid. Static analysis is done without running an application. Some examples of static features include, permissions, API calls which can be extracted from the AndroidManifest.xml file. Dynamic analysis deals with features that were extracted from the application while running, including network traffic, battery usage, IP address. The third type of analysis is hybrid analysis which combines the features from static and dynamic techniques. However, with the update attack, it was concluded that it is best to compare old version product with the new version so as to determine if the exist concerns to be addressed.
Saif D. et all also sought to develop a program that would help detect and classify malicious applications in android devices. From 2010 to 2018 many different anti malware applications were developed. These app used either static, dynamic, or both ways to identify malware. Finally,

in 2018 Hui-Juan Zhu et developed an 88.26% accuracy, 88.40% sensitivity and 88.16% precision software to detect malware.
Alatwi A. H et al utilized features of benign applications to detect malware by relating the feature requested to the common feature of its classification. It therefore proposed a category-based machine learning classifier to enhance performance of classification models knowing that apps within the same category have a common set of features such as permissions, APIs, intents and filters among other features.
Huang, H. et al sought to develop an android malware detection system that could ensure that the cyber security of android smartphones was maintained. Chen T. et al Observed that malicious people were easily able to inject malware into android systems successfully without the malware being detected through the use of code obfuscation.
S. Wang et al perform text-like segmentation and vectorization on URLs to analyze malware using network traffic. The work utilized a multi-view neural network to implement deep and broad discriminative feature learning that addresses the feature selection difficulty in malware detection via network traffic. To evaluate different influential factors, multi-group experiments were performed on malware detection mode.
Ganesh M. et al also proposed a convolution neural network based android malware detection system and identified its main challenge as the high false detection rate. Wang W. et al proposed a hybrid model that include the deep auto encoder and the convolutional neural network deep learning algorithms to detect android malware.
Park et al (2020) show malware and adware used to infect self-driving vehicle that use the Android OS. This approach used a machine learning based intrusion detection module combined with a machine learning algorithm where accuracy and speed are critical.

3.0 Deep Learning Algorithms for Malware Detection

Six deep learning algorithms discussed by literature as possible solutions to deep learning for android malware detection and classification are: deep neural network (DNN), restricted Boltzmann machine (RBM), convolutional neural network (CNN), deep belief network (DBN), recurrent neural network (RNN), and the deep autoencoder.
When evaluating the strengths and weaknesses of these deep learning algorithms, The reading considers the accuracy of the algorithm, the computational power required to support the running of the algorithm, the time spent training the algorithm, the size of data sets used to train the algorithm, and the ability of the algorithm to be used under various circumstances or applications. The reading also considers the level of difficulty when training the algorithm, and the number of variations that exist for a given algorithm.
The major strength of the DNN is that it has been used in different applications. However, its weakness is that its training consumes a lot of time. The major strength of the RBM is that it can be used as a feature extractor for the other algorithms. However, its weakness is that its training is time consuming, and it also requires a lot of computational power. CNN's major strength is that it has many variations. However, it needs a large data set to train. The major strength of DBN is that it has a layer by layer learning approach. However, this gives it the weakness of requiring a lot of computational power, and time consumption during training. The major strength of the RNN is that it can remember serial events, however, the learning process has the problem of

vanishing gradient. The major strength of the deep autoencoder is that it has numerous variations and can be used together with other algorithms to create a hybrid. However, it lacks the ability to determine pertinent data.

According to authors, the review of the different deep learning algorithms that can be used or that have been proposed as solutions to deep learning malware detection in android devices was the first of its kind, although there are other related reviews. The aim was to fill the existing gaps in the literature regarding the use of deep learning to detect and classify malware in android devices.

4.0 Methodology

The data was collected by determining the relevant information from the literature, then listing the keywords to be used in the search process for the repositories. After the search results, exclusion criteria are applied to confine the review of the pertinent papers. An inclusive list of reviews is made from the collected papers. The search keywords are summarized as deep learning malware detection and analysis. Other learning strategies applied include the convolutional neural network in Android malware analysis; deep belief network in Android malware analysis; recurrent neural network in Android malware analysis. The search database repository probe finds significant publications such as Web Knowledge and Science Direct. Among the criteria used for exclusion to find only relevant results include finding papers published in a non-English language, those published in their final versions, duplicated papers, and those using deep-learning for malware detection in Windows.

5.0 Review of Literature Analysis

Various studies have been conducted in regards to malware detection for android devices. A static analysis using a malware identification system utilizing the deep learning method attained an accuracy of 97.4%. This approach is designed to seek permission from other apps and converting permission into image files. With an accuracy of 93% on unbalanced data, it is evident that deep learning offers an exact and extensible solution for Android malware since it relies on patterns to determine the malware.

The malware detection system, DeepFlow builds on data streams in malignant apps that may contrast the original ones. The contrasts are used to distinguish novel apps using a deep learning model. An alternative method would be to apply API calls that occurred similarly to the small code. A different approach to detecting malware is through CNN. In this approach, the features are gained from raw data and malware signatures are eliminated.

Other detection models include the MalDozer that depends on an artificial neural network to identify the malware. The color compounded convolution neural network-based AMD depends on color representation to translate images and decode the malware. The DeepRefiner malware characterization uses deep neural networks with several hidden layers to automate feature extraction. This is also seen in DroidDeep.

Other models of malware detection include dynamic analysis such as natural language processing techniques, and Deep4MalDroid; and the hybrid analysis such as Droidsec and DDfender.

Hou et al (2016), carried out a study whose purpose was to propose a deep learning model that could detect malware in android devices that was difficult to detect in other devices. The proposed method used hybrid analysis, since it included features of both static analysis and dynamic analysis. It used the extracted Linux kernel system calls. The name of the proposed model was Deep4MakDroid. The experimental results of the model however showed that it performed relatively less accurately compared to other models. The experimental results showed that the proposed model was able to outperform all the other models which it was compared with, based on efficiency and accuracy.

Yuan et al (2014), conducted a study to devise a machine learning method that could detect android malware devices. The researchers used a machine learning method that utilizes more than 200 features, using both static and dynamic analysis. The authors appreciated the fact that android malware were being developed at a rate that the traditional detection methods could not keep up. Experimental results of the proposed method showed that the proposed malware detection model had an accuracy level of 96%.

Kim et al (2018), conducted a study whose objective was to propose a novel framework for android malware detection. The researchers used a multimodal approach, which they touted as the first of its kind, and that would revolutionize android malware detection. The research used the deep neural networks, and the model was tested on a sample size of 41,260 applications. The experimental results of the model however showed that it performed relatively less accurately compared to other models.

Su et al (2016), sought to create a model that would be able to use machine learning technology to detect new software that was not yet registered in databases, and determine whether it was benign or not. The researchers evaluated the proposed model (DroidDeep) against a data set of a total of 7972 samples. Half of the sample was made of malicious software. The experimental results showed that the model had a high runtime efficiency and a detection accuracy of 99.4%.

McLaughlin et al (2017), also conducted a study to propose a deep learning method for android malware detection. The proposed model used a convolutional neural network deep learning algorithm in order to create the proposed model. The convolutional neural network used static analysis on raw opcode sequence to detect android malware. The results of the study showed that the model was able to scan large number of files accurately and efficiently on a GPU.

Karbab et al (2017), conducted a study to propose a model for detecting android malware using deep learning on API methods sequence. The proposed model used an API methods sequence to classify applications and in the process identify malware applications. The name of the proposed model was MalDozer. The researchers used sample sizes that consisted of datasets of between 1k-33k malware, and 38k benign apps to detect malware. The experimental results showed that the proposed model had an accuracy of 96-99% when classifying software. The experimental results also showed that the model had a false positive rate of 0.062% - 2%, on the tested data set.

Li D. et al (2018), conducted a study to determine a model that would follow a fine-grained android malware detection. The proposed model followed a static analysis, and compared it with other methods in its experimental stages. The experimental results showed that the proposed model has an accuracy level of 97%, and a false positive rate of 0.1%.

Pektaş et al (2020), conducted a study with the objective to propose a method for effective android malware detection using API call graph embedding. The researchers used the API call

graph to represent the possible execution paths. Also, to achieve the objective, the researchers combined all parameters to ensure the best combination of hyper-parameters. The experimental results of the proposed model had an accuracy level of 98.86%, an accuracy of 98.65% in F-measure, 98.47% in recall, and 98.84% precision.

Li W. et al (2018), in their research proposed an android malware detection approach using weight adjusted deep learning. The algorithm that was used to develop the model was deep neural networks. The model used a characterization and identification approach. However, the proposed model had an accuracy level which was relatively lower than other methods proposed by literature. The accuracy level of the proposed model was 90%.

Xu et al (2018) conducted a study to propose an android malware detection system based on deep learning that uses CFG and DFG. In the study, the algorithm used was convolutional neural networks. In order to detect the malware applications, the proposed model used semantic graphs to characterize android applications. In order to experiment the proposed model, the researchers used arvin, Drebin, VirusShare and ContagioDump datasets. The experimental results showed that the proposed model performed better than some previous studies and many anti-virus tools gathered in VirusTotal.

Saif et al (2018), conducted a study to propose a model that they could use to detect and classify malicious applications in android devices. The research used the deep belief network to create the model. The experimental results of the proposed model showed an accuracy level of 99.1%.

Chen et al (2019), also conducted a study in order to create a novel Android malware detection system that would detect malware that was novel and not yet recorded in databases. The research used the deep belief network in order to create the proposed model. The experimental data used had a total of 15,000 applications. The results of the experiment showed that the proposed model had an accuracy level of 99.10% - 99.40%, in detecting malicious software.

Su et al (2020), also designed a model to detect malware in android devices using a deep belief network. The proposed model collected data from 11 different static behavioral characteristics. The researcher then experimented with the performance of the proposed model comparing with others already proposed by previous studies. The results of the study showed that the proposed system had an accuracy level of 99.4%.

Wang et al (2019), conducted a study to propose an android detection malware software. The researchers approached the research problem by developing a hybrid model that was a hybrid between autoencoder and convolutional neural network. The deep learning algorithm was trained used a total of 23,000 software, where 13,000 of these were malicious software. The experimental results showed that the proposed model had an accuracy level that was 5% higher than other models that used SVM.

Ganesh et al (2017), proposed a model that used a convolutional neural network-based android malware detection system. The researchers sought to use the algorithm to classify different software and therefore detect the malicious software. In order to determine the efficacy of the proposed model, the researcher used data that had 2500 android software. The experimental results of the proposed model had an accuracy level of 93%.

Hasegawa et al (2018), also conducted a study to propose a model that would be used to detect android malware, while at the same time overcoming the problem of having to deal with the limited computation power of most android devices. To overcome this problem, the researcher proposed a one-dimensional convoluted neural network. The researchers then experimented

with the performance of the proposed model using a data set of 7,000 software, where 5,000 of them were android malware. The experiment results showed that the proposed model had an accuracy level of 95.40-97.04%.

Markel et al (2014), conducted research with the aim of proposing a model that would use the deep neural network algorithm to detect malware in android devices and to determine the efficiency and accuracy of using a deep neural network to detect such malware. The researchers then experimented with the study and compared its performance with the performance of other models that had been previously discussed by literature. The proposed algorithm had an accuracy level of 95.31%. This accuracy level was higher than the other models that it was compared to in the specific research.

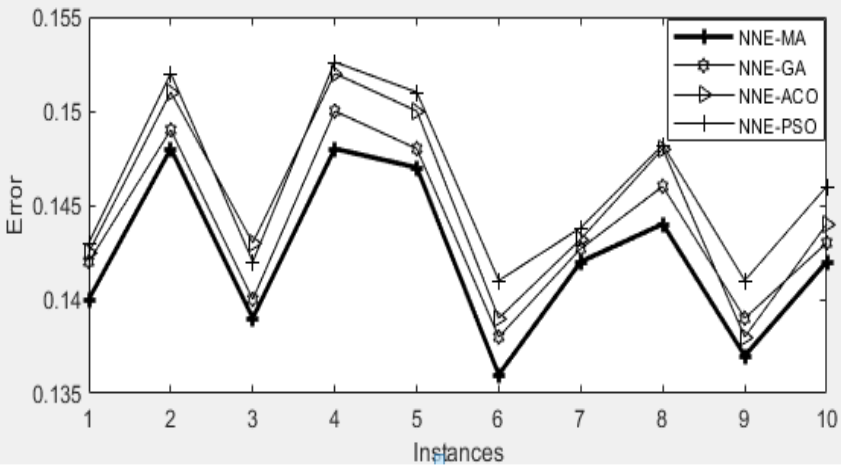
Chen et al (2019), also conducted a study with the aim of proposing a model that would solve the problem of malware detection of most novel malware evading detection by using complex features. The researchers proposed a model that uses the deep neural network algorithm to detect Android malware. The proposed model was experimented on using a data set that had 41,260 samples. The accuracy level of the proposed model was higher than the other models. It had an accuracy level of 99.10% - 99.40%.

Martín et al (2017), also set out to propose a model that would detect android malware using the deep neural networks architecture. The proposed model would classify and identify the malware, based on the extract features of the malware. The accuracy of the proposed model was improved by using a new parameter architecture and genetic algorithm. The researchers then conducted a study to determine the performance of the proposed model. The experimental results showed that the proposed model had an accuracy level of 91%.

Mohammed et al (2020) made DL-Droid which is a deep learning system to detect malicious Android applications. It uses dynamic analysis using stateful input generation. They did tests on over 30,000 applications. Experiments were done to compare the detection performance and code coverage of the stateful input generation method with the commonly used stateless approach using the deep learning system. According to (2) DL-Droid can achieve up to 97.8% detection rate using dynamic analysis only and 99.6% detection rate with dynamic and static analysis.

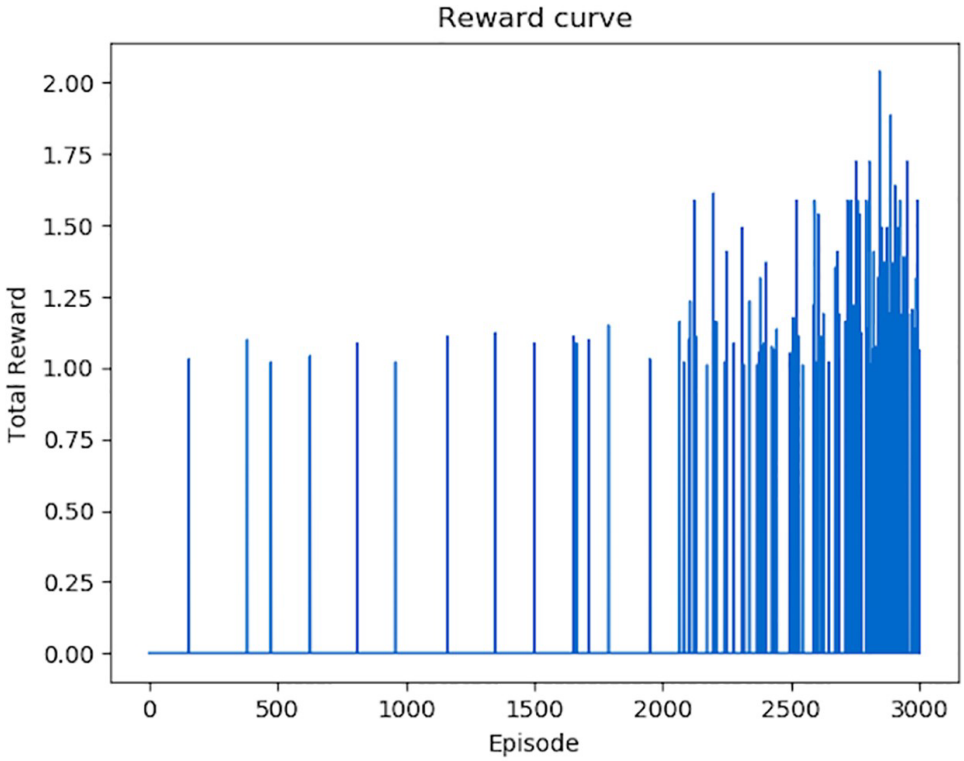
Dali et al (2017) developed DeepFlow, the results show a high detection F1 score of 95.05%, outperforming traditional machine learning-based approaches, which reveals the advantage of deep learning technique in malware detection.

The figure below depict the work done by Akandwanaho and Kooblal in 2019 at *The African Journal of Information and Communication (AJIC)* using the neural network ensemble and metric algorithm (NNE-MA) in combination with other proven techniques at the time to include the generic algorithm (GA), ant colony optimization (ACO), and particle swarm optimism (PSO) algorithms. NNE-MA showed to produce less errors than the existing techniques at the time.



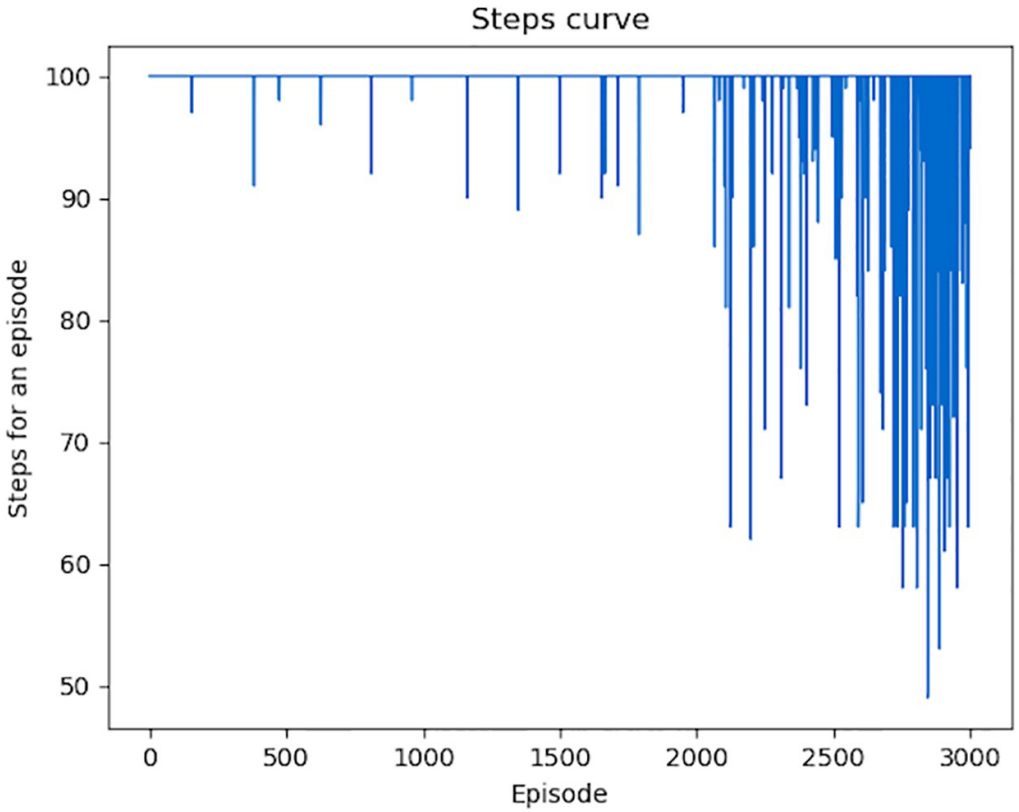
Ref - Akandwanaho, S. M. , & Kooblal, M. (2019)

Zhong et al discussed Multi-level deep learning system (MLDLS) as mapped out in five phases that heavily involve MapReduce. It detects malware with high accuracy and remove anomalies. In 2019, MDLS excelled in true and false positive rates, construction and comparison time when experimented. The figure below shows a graphical representation of the result



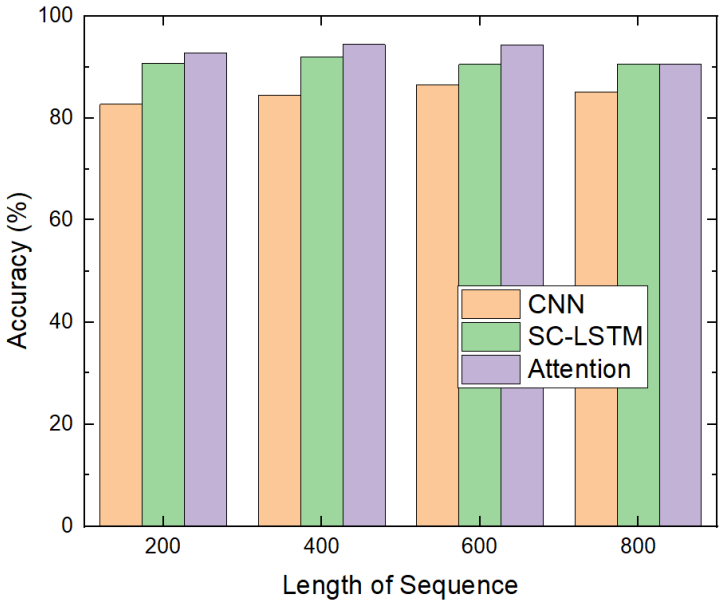
Ref – Chen et al (2019)

Fang et al (2020) conducted an experiment using DeepDetectNet vs RLAttackNet where Portable Executive (PE) malware detection and the use of Generative Adversarial Network (GAN) to upgrade PE malware detection, and by combining both; levels of security were astonishing. The figure below is a representation of the outcome of this experiment



Fang et al (2020)

Choi et al (2020) discussed a deep learning-based model utilizing an attention based method, the accuracy of attention out performed long short-term memory (SC-LSTM) and Convolutional neural network (CNN), with an accuracy of 12% & 5% higher. They presented the chart below showing accuracy as a function of length of sequence.



Ref - Choi et al (2020)

Ye et al (2018) found the results of using a heterogeneous deep-learning frame work that includes an AutoEncoder and a layer of associative memory to detect unknown malware by way of pre-training and fine-tuning. The figure below shows how malwares are identified

Comparisons with different anti-malware scanners

Malware	K	M	S	T	VT	DeepAM
M1	×	√	×	×	5/56	√
M2	√	×	√	×	7/56	×
M3	×	×	×	×	12/56	√
M4	×	√	×	×	9/56	√
M5	×	×	×	×	16/56	√
M6	×	×	×	√	13/56	√
M7	×	×	×	×	5/56	√
M8	×	×	×	×	3/56	×
M9	√	×	√	×	10/56	√
⋮	⋮	⋮	⋮	⋮	⋮	⋮
M500	×	√	√	√	19/56	√
TP	407	365	418	358	453	489
TPR	81.4%	73.0%	83.6%	71.6%	90.6%	97.8%

X/*Y* for VT denotes VirusTotal's detection ratio where *X* out of *Y* anti-malware vendors detect the file as malicious. We here assume that if there are ≥1/3 (i.e., 19/56) of the anti-malware vendors detecting the file as malicious, then it will be recognized as malware by VT

Ref - Ye et al

The following table shows a summary of the different algorithms used and their accuracy levels

Researchers	**Methodology and or Deep Learning Algorithm Used**	**Findings/Accuracy**
Hou, Saas, Chen and Ye (2016)	Dynamic analysis, using the extracted Linux kernel system calls.	The proposed model (Deep4MakDroid) was able to outperform other malware detection software.
Yuan, Lu, and Wang (2014)	Used a machine learning method that utilizes more than 200 features, using both static and dynamic analysis.	The malware detection had an accuracy level of 96%.
Kim, Kang, Rho, Sezer and Im (2018).	- Used a multimodal approach and was the first of its kind. - Performance evaluated on 41,260 samples. - Used deep neural networks models	The performance and accuracy were relatively lower compared to other models

Su, X., Zhang, D., Li, W., & Zhao, K. (2016)	- Used a detection approach (DroidDeep). - The performance evaluated on a total of 7972 samples	- High runtime efficiency - A detection accuracy of 99.4%
McLaughlin, N., Martinez del Rincon, J., Kang, B., Yerima, S., Miller, P., Sezer, S., ... & Joon Ahn, G. (2017)	- Used the Convolutional neural network - Based on static analysis on raw opcode sequence.	- File was able to scan large number of files accurately and efficiently on a GPU.
Karbab, E. B., Debbabi, M., Derhab, A., & Mouheb, D. (2017)	- API methods sequence - Proposed MalDozer, which uses sequence classification - Used various datasets of between 1k-33k malware, and 38k benign apps to detect malware	- Has an accuracy of 96-99% when classifying software - Has a false positive rate of 0.062% - 2%, on the tested data set.
Li, D., Wang, Z., & Xue, Y. (2018)	Deep Neural Network	- Has an accuracy level of 97% - Has a false positive rate of 0.1%
Pektaş, A., & Acarman, T. (2020).	- Used the API call graph to represent the possible execution paths. - Combined all parameters to ensure the best combination of hyper-parameters	- Results from the experiments show an accuracy level of 98.86%, an accuracy of 98.65% in F-measure, 98.47% in recall, and 98.84% in precision.
Li, W., Wang, Z., Cai, J., & Cheng, S. (2018).	- Deep neural networks - Used a characterization and identification approach - Experimented on the efficacy of the model using 237 features	- Accuracy relatively lower than other methods proposed by literature - Accuracy level was 90%
Xu, Z., Ren, K., Qin, S., & Craciun, F. (2018)	- The algorithm used was convolutional neural networks - Used semantic graphs to characterize android applications - Used arvin, Drebin, VirusShare and ContagioDump datasets to	- Performs better than Yeganeh Safaei et al.'s approach, Allix et al.'s approach, Drebin and many anti-virus tools gathered in VirusTotal,

	experiment the proposed model	
Markel, Z., & Bilzor, M. (2014)	Deep Neural Networks	95.31
Martín, A., Fuentes-Hurtado, F., Naranjo, V., & Camacho, D. (2017)	Deep Neural Networks	91%
Hasegawa, C., & Iyatomi, H. (2018)	Convolutional Neural Network	95.40-97.04%
Wang, W., Zhao, M., & Wang, J. (2019).	Convolutional Neural Network	5% higher than other models that used SVM
Ganesh, M., Pednekar, P., Prabhuswamy, P., Nair, D. S., Park, Y., & Jeon, H. (2017)	Convolutional Neural Network	93%
Su, X., Shi, W., Qu, X., Zheng, Y., & Liu, X. (2020)	Deep Belief Network	99.4%
Chen, T., Mao, Q., Lv, M., Cheng, H., & Li, Y. (2019)	Deep Neural Networks	99.10% - 99.40%
Saif, D., El-Gokhy, S. M., & Sallam, E. (2018)	Deep Belief Network	99.1%
Mohammed K. Alzaylaee, Suleiman Y. Yerima, Sakir Sezer (2020)	Dynamic analysis using stateful input generation	97.8% - 99.6%
Dali Zhu, Hao Jin, Ying Yang, D. Wu and Weiyi Chen (2017)		95.05%

5.1 Discussion

It is evident there have been several research into the application of deep learning in the detection of malware in android devices. The adoption of this technique however is dependent on several factors to include time, complexity of need and cost. For example, the decision to implement static, dynamic or hybrid analysis approach is solely dependent on the need of the administrator and complexity of the platform involved. Various algorithms were presented with different strengths and weaknesses. These algorithms also come with different accuracy levels. Since deep learning uses artificial intelligence by training models, available dataset becomes another critical component to be put into consideration when selecting the algorithm to implement. Time of training is also important to consider as some models expend more time during training compared to others. The one thing that is certain is the fact that this scheme has proven to be astronomically more effective in detecting malware when compared to traditional malware detection software.

5.2 Challenges and Open Issues

A number of open issues exist with the most technical being availability of enough datasets for the training of any model to be implemented. It is also of utmost importance to deal with the time involved in the training of these models. Concept drift is a concern as it applies to the rapid growth we are experiencing, security issues with deep learning, and data privacy preservation.

6.0 Conclusion

With this review of the use of deep learning in Android malware detection, a comparison of available work was presented. This review identified knowledge gaps and highlight challenges, and open issues that should be considered in future work. It was shown that deep learning methods are subject to various adversarial attack, concept drift remains a challenge, and that static analysis have dominated the existing work. Generally, the result of this work can aid to promote research in Android malware detection and maybe answer the question of which model is the best to implement.

References

[1] A. Naway and Y. Li, "A Review on The Use of Deep Learning in Android Malware Detection," p. 15.

[2] B. Baskaran and A. Ralescu, "A Study of Android Malware Detection Techniques and Machine Learning," p. 9, 2016.

[3] Saif, D., El-Gokhy, S. M., & Sallam, E. (2018). Deep Belief Networks-based framework for malware detection in Android systems. *Alexandria engineering journal*, *57*(4), 4049-4057.

[4] Alatwi H. A, Oh T. , Fokoue E., and Stackpole B., "Android Malware Detection Using Category-Based Machine Learning Classifiers," in *Proceedings of the 17th Annual Conference on Information Technology Education*, New York, NY, USA, Sep. 2016, pp. 54–59, doi: 10.1145/2978192.2978218.

[5] Chen, T., Mao, Q., Lv, M., Cheng, H., & Li, Y. (2019). DroidVecDeep: Android Malware Detection Based on Word2Vec and Deep Belief Network. *TIIS*, *13*(4), 2180-2197.

[6] Huang, H., Chen, K., Ren, C., Liu, P., Zhu, S., & Wu, D. (2015, April). Towards discovering and understanding unexpected hazards in tailoring antivirus software for android. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security* (pp. 7-18).

[7] S. Wang *et al.*, "Deep and Broad Learning Based Detection of Android Malware via Network Traffic," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, Banff, AB, Canada, Jun. 2018, pp. 1–6, doi: 10.1109/IWQoS.2018.8624143.

[8] Ganesh, M., Pednekar, P., Prabhuswamy, P., Nair, D. S., Park, Y., & Jeon, H. (2017, July). Cnn-based android malware detection. In *2017 International Conference on Software Security and Assurance (ICSSA)* (pp. 60-65). IEEE.

[9] Wang, W., Zhao, M., & Wang, J. (2019). Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network. *Journal of Ambient Intelligence and Humanized Computing*, *10*(8), 3035-3043.
[10] Chen, T., Mao, Q., Lv, M., Cheng, H., & Li, Y. (2019). DroidVecDeep: Android Malware Detection Based on Word2Vec and Deep Belief Network. *TIIS*, *13*(4), 2180-2197.
[11] Hsien-De Huang, T., & Kao, H. Y. (2018, December). R2-D2: color-inspired convolutional neural network (CNN)-based android malware detections. In *2018 IEEE International Conference on Big Data (Big Data)* (pp. 2633-2642). IEEE.
[12] Martín, A., Fuentes-Hurtado, F., Naranjo, V., & Camacho, D. (2017, June). Evolving deep neural networks architectures for android malware classification. In *2017 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1659-1666). IEEE.
[13] Markel, Z., & Bilzor, M. (2014, October). Building a machine learning classifier for malware detection. In *2014 Second Workshop on Anti-malware Testing Research (WATeR)* (pp. 1-4). IEEE.
[14] Hou, S., Saas, A., Chen, L., & Ye, Y. (2016). "Deep4MalDroid: A deep learning framework for Android malware detection based on Linux kernel system call graphs," Proc. - 2016 IEEE/WIC/ACM Int. Conf. Web Intell. Work. WIW 2016, pp. 104–111, 2017.
[15] Li, W., Wang, Z., Cai, J., & Cheng, S. (2018, March). An android malware detection approach using weight-adjusted deep learning. In *2018 International Conference on Computing, Networking and Communications (ICNC)* (pp. 437-441). IEEE.
[16] Li, D., Wang, Z., & Xue, Y. (2018, May). Fine-grained android malware detection based on deep learning. In *2018 IEEE Conference on Communications and Network Security (CNS)* (pp. 1-2). IEEE.
[17] Hou, S., Saas, A., Chen, L., & Ye, Y. (2016, October). Deep4maldroid: A deep learning framework for android malware detection based on linux kernel system call graphs. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW)* (pp. 104-111). IEEE.
[18] Karbab, E. B., Debbabi, M., Derhab, A., & Mouheb, D. (2017). Android malware detection using deep learning on API method sequences. *arXiv preprint arXiv:1712.08996*.
[19] Kim, T., Kang, B., Rho, M., Sezer, S., & Im, E. G. (2018). A multimodal deep learning method for android malware detection using various features. *IEEE Transactions on Information Forensics and Security*, *14*(3), 773-788.
[20] McLaughlin, N., Martinez del Rincon, J., Kang, B., Yerima, S., Miller, P., Sezer, S., ... & Joon Ahn, G. (2017, March). Deep android malware detection. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy* (pp. 301-308).
[21] Pektaş, A., & Acarman, T. (2020). Deep learning for effective Android malware detection using API call graph embeddings. *Soft Computing*, *24*(2), 1027-1043.
[22] Su, X., Zhang, D., Li, W., & Zhao, K. (2016, August). A deep learning approach to android malware feature learning and detection. In *2016 IEEE Trustcom/BigDataSE/ISPA* (pp. 244-251). IEEE.
[23] Xu, Z., Ren, K., Qin, S., & Craciun, F. (2018, November). CDGDroid: Android malware detection based on deep learning using CFG and DFG. In *International Conference on Formal Engineering Methods* (pp. 177-193). Springer, Cham.

[24] Yuan, Z., Lu, Y., Wang, Z., & Xue, Y. (2014, August). Droid-sec: deep learning in android malware detection. In *Proceedings of the 2014 ACM conference on SIGCOMM* (pp. 371-372).
[25] Mohammed K. Alzaylaee, Suleiman Y. Yerima, Sakir Sezer, DL-Droid: Deep learning based android malware detection using real devices, Computers & Security, Volume 89, 2020, 101663, ISSN 0167-4048
[26] Dali Zhu, Hao Jin, Ying Yang, D. Wu and Weiyi Chen, "DeepFlow: Deep learning-based malware detection by mining Android application for abnormal usage of sensitive data," 2017 IEEE Symposium on Computers and Communications (ISCC), Heraklion, 2017, pp. 438-443, doi: 10.1109/ISCC.2017.8024568.
[27] Akandwanaho, S. M. , & Kooblal, M. (2019). Intelligent malware detection using a neural network ensemble based on a hybrid search mechanism. *The African Journal of Information and Communication (AJIC),* 24, 1-21. https://doi.org/10.23962/10539/28660
[28] Zhong, W., & Gu, F. (2019). A multi-level deep learning system for malware detection. *Expert Systems with Applications*, 133, 151–162. https://doi-org.blume.stmarytx.edu/10.1016/j.eswa.2019.04.064
[29] Fang Y, Zeng Y, Li B, Liu L, Zhang L. DeepDetectNet vs RLAttackNet: An adversarial method to improve deep learning-based static malware detection model. *PLos ONE*. 2020;15(4):1. Accessed November 17, 2020 http://search.ebscohost.com.blume.stmarytx.edu:2048/login.aspx?direct-true&db=edb&AN=142872806&site=eds-live%scope=site
[30] Choi, Sunoh, et al. “Attention-Based Automated Feature Extraction for Malware Analysis.” *Sensors (14248220)*, vol. 20, no. 10, May 2020, p. 2893. *EBSCOhost*, search.ebscohost.com/login.aspx?direct=true&db=edb%AN=143762309&site=eds-live&scope=site.
[31] Ye, Yanfang, et al. “DeepAM: A Heterogenous Deep Learning Framework for Intelligent Malware Detection.” *Knowledge & Information Systems*, vol. 54, no. 2, Feb. 2018, p. 265. *EBSCOhost*, search.ebscohost.com/login.aspx?direct=true&db&AN=127498065&site=eds-live&scope=site.
[32] Park, Seunghyun, and Jin-Young Choi. “Malware Detection in Self-Driving Vehicles Using Machine Learning Algorithms.” *Journal of Advanced Transportation*, Jan. 2020, pp. 1–10. *EBSCOhost*, doi:10.1155/2020/3035741.