Droid-NNet: Deep Learning Neural Network for Android Malware Detection

Mohammad Masum
Analytics and Data Science Institute
Kennesaw State University
Kennesaw, USA
mmasum@students.kennesaw.edu

Hossain Shahriar
Department of Information Technology
Kennesaw State University
Marietta, USA
hshahria@kennesaw.edu

***Abstract*— Android, the most dominant Operating System (OS), experiences immense popularity for smart devices for the last few years. Due to its' popularity and open characteristics, Android OS is becoming the tempting target of malicious apps which can cause serious security threat to financial institutions, businesses, and individuals. Traditional anti-malware systems do not suffice to combat newly created sophisticated malware. Hence, there is an increasing need for automatic malware detection solutions to reduce the risks of malicious activities. In recent years, machine learning algorithms have been showing promising results in classifying malware where most of the methods are shallow learners like Logistic Regression (LR). In this paper, we propose a deep learning framework, called Droid-NNet, for malware classification. However, our proposed method Droid-NNet is a deep learner that outperforms existing cutting-edge machine learning methods. We performed all the experiments on two datasets (Malgenome-215 & Drebin-215) of Android apps to evaluate Droid-NNet. The experimental result shows the robustness and effectiveness of Droid-NNet.**

Keywords—neural network, android malware, android security

I. Introduction

Malware is malicious software (e.g. viruses, ransomware, trojan horses, and spyware) that can damage or execute harmful actions on devices [2]. Android is one of the most accepted OS for smart devices like phones, tablets, and other mobile devices. Due to its popularity and open characteristics, Android is prone to malware attacks, which can cause devastating effects such as stealing information, corrupting files, infecting entire network of devices [1]. Therefore, malware poses a major security threat to financial institutions, businesses, and individuals.

The number of malware threats on Android-based smart devices are increasing exponentially and the newly created malware has become more sophisticated and variants. Hence, traditional malware detection techniques such as signature-based detection, heuristic detection or behavior-based detection are not adequate to combat malicious software [2].

Machine learning algorithms have been showing promising results in classifying Android malware. The algorithms can overcome the limitations of traditional detection methods and provide a rewarding accuracy score. Machine learning approaches like Support Vector Machine (SVM), Logistic Regression (LR), and Decision Tree (DT) were previously proposed for malware detection [5].

Neural networks are currently widely used for many applications due to the capability of highly non-linear systems and flexibility in architecture design. In this paper, we propose a deep neural network framework named Droid-NNet for Android malware detection. Our contributions include: (1) We conduct a comprehensive assessment with rigorous experimental setting to asses Droid-NNet performance with two publicly available real-world Android application datasets, and (2) Droid-NNet provides high weighted F-beta score, high true positive rate and low false positive rate based on deep neural network architecture which suggests that detecting Android malware using deep learning technique is promising.

The rest of the paper is organized as follows: In Section II, we introduce the related work of Android malware detection. Section III describes the methodology of our proposed method Droid-NNet along with the other three classifiers that are implemented in this paper. The experimental setting and results are explained in Section IV. Finally, Section V concludes the paper.

II. Background & Related Work

Traditional detection techniques have been applied for classifying Android malware. Signature-based detection is the most widely used anti-malware system. It identifies a malware instance by searching specified byte sequences (called signatures) into an object to investigate matching with known signatures from blacklisted malicious programs. The detection method is not effective against "zero-day attacks" as the system is formed based on known malware signatures [3]. A signature-based detection method was proposed to Android malware detection that leverages signature matching algorithms. An extension of signature-based method was proposed that combines anomaly-based and signature-based mechanisms. The combined approach achieved 96% accuracy in classifying malicious apps by experimenting on three different data sets [8].

To overcome the limitations of signature-based detection, behavior-based malware detection was proposed [4]. The behavior-based technique analyzes the behavior of a program when it is executing and defines the program as malware if it does not execute normally. However, the method affects the system's performance, requires more space, and generates many false positives and false negatives [3]. Behavior-based Android malware detection method MADAM was proposed which simultaneously analyzes and correlates features at different levels. MADAM achieved 95% accuracy in classifying malware [8].

978-1-7281-0858-2/19/$31.00 © 2019 IEEE

Addressing the constraints of the traditional methods, researchers have proposed machine learning algorithms for malware classification. A linear SVM was applied to detect Android malware. A set of 32 features that are highly related to targeted malware are used in this study and achieved an F-measure of 0.954 [13]. A multilevel classifier fusion approach, DroidFusion, was proposed for Android malware detection. DroidFusion contains two layers wherein the upper layer, a regular classifier is used, and a ranking based classifier was then applied to reassign the label of test data. DroidFusion experimented with four different datasets including Malgenome-215 and Drebin-215 datasets. For Malgenome-215 data, DroidFusion achieved 0.9840 weighted F-measure score while 0.9872 weighted F-measure was obtained for the Drebin-215 dataset.

Neural Network for Android Detection of Malware (NADM) was proposed leveraging two fully connected hidden layers. NADM was implemented on large-scale data that contains more than 1 million samples and achieved an average 90% accuracy in detecting malware [16]. Random Forest (RF) classifier and deep neural network with three different architecture (2, 4, and 7 layers) implemented on a dataset where 11,308 malicious files were collected from the Malacia project and 2,819 benign files were collected from "virustotal.com" [14]. The paper also presented four more different experimentation with varying the number of features which were extracted using autoencoder with different threshold approach. Irrespective of feature sets, RF outperformed the deep network and obtained 99% accuracy. DeepDetector, an Android malware detection method based on deep learning, was proposed that can detect malicious applications and fine-grained malware families at the same time. DeepDetector was tested with varying hidden layers and a different number of neurons and a maximum 94% F1 score was obtained in malware classification.

Our proposed method Droid-NNet, a neural network framework, was optimized with different parameters and hyperparameters and experimented with two real-world Android applications datasets. The experimental results show the robustness and effectiveness of Droid-NNet.

III. Methodology

A. Support Vector Machine

SVM is a well-known supervised learning technique to analyze high-dimensional data. SVM searches for an optimal hyperplane in the input space that categorizes two classes given training data. Therefore, the hyperplane is used to classify new data [6]. Fig. 1 illustrates a hyperplane for SVM that separates two classes.

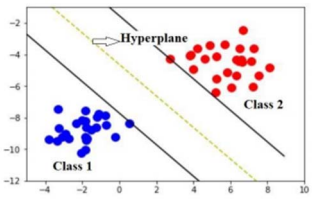

Fig. 1: Hyperplane for an SVM trained with samples from two classes

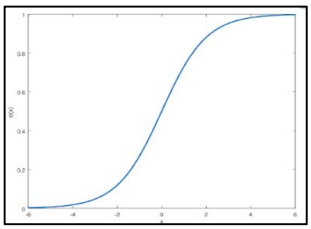

Fig. 2: An example of sigmoid function

B. Decision Tree

Decision Tree is a well-known supervised machine learning technique for classification. It builds a classification model in the shape of a tree structure through a process known as binary recursive partitioning [7]. It iteratively splits the data into smaller and smaller subsets (branches) until each of the branches achieves homogeneous partitions. Therefore, it finally creates a tree with decision nodes and leaf nodes where the decision nodes contain two or more branches and leaf node assigns a class or decision.

C. Logistic Regression

Logistic regression is a classical classifier of supervised learning. It utilizes the sigmoid function to squeeze the output of a linear equation between 0 and 1. Thus, the output of logistic regression can be used to predict the probability of a class [6]. Fig. 2 shows an example of a sigmoid function.

D. Neural Network

At present, neural networks are widely used for many applications due to the capability of highly non-linear systems and flexibility in architecture design. The neural network's basic architecture contains input layers, one or more hidden layers, and output layers where each of the layers includes a certain number of neurons. Weighted linear combination of neurons of a layer is computed and then used as input to another neuron in the succeeding layer. To capture the non-linearity of the data, a non-linear function, called activation function, can be applied to the weighted sums of neurons. All the weights of a neural network are set to random values at the initial stage of training. Data is fed into the input layer of the network, then it travels through the hidden layers, and finally output is produced in the output layer. The network continually updates the weights applying backpropagation based on the output and desired target of the neural network. The network consequently reduces the error between the output and target in each iteration [17]. In the process, a loss function is used to

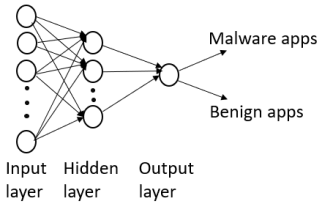

Fig. 3: Architecture of Droid-NNet

calculate the error of the network and the error is minimized by applying optimization function during backpropagation.

In this paper, we propose a neural network framework named Droid-NNet. Fig. 3 shows the architecture of Droid-NNet which is a neural network containing three layers: input layer, hidden layer, and output layer. A threshold is applied to the output layer to classify the instances as malware apps or benign apps. The input layer contains 215 neurons (number of features of samples), the hidden layer contains 25 neurons and the output layer includes only one neuron since the problem is a binary classification. We applied binary cross-entropy as loss function and Adaptive Moment Estimation (Adam) optimizer for calculating error and updating the parameters.

IV. Experiment & Results

A. Dataset specification

We performed all experiments on two datasets (Malgenome-215 & Drebin-215) of Android apps to evaluate Droid NNet. Each of the dataset's details are shown in Table I. Drebin-215 dataset is publicly available and Malgenome-215 dataset is collected from the supplementary section of [10]. Malgenome-215 dataset has a total of 3,799 app samples, where 2,539 and 1,260 are benign and malware samples, respectively from the Android malware genome project [11]. The Drebin-215 dataset consists of 15,036 samples of apps in which 9,476 are benign and the remaining 5,560 are malware from the Drebin project [12]. Both datasets contain 215 features.

B. Model evaluation metrics

Both datasets we utilized in this paper are unbalanced. The proportion among benign and malware samples in the Malgenome-215 dataset is approximately 66% and 33% respectively. In the Drebin-215 dataset, the ratio of benign and malware samples is approximately 63%: 37%. Therefore, we ought not to consider the "accuracy" metric to assess the performance of the models. Thus, the following performance measurements are considered in the assessment of the models [10].

1. *TPR (True Positive Rate / Recall):* The proportion of correctly identified malware apps (TP) to the total number of malware applications (TP+FN). TP (True Positive) is the quantity of correct predictions while FN (False Negative) is the amount of malware misclassified.

$$TPR = \frac{TP}{TP + FN}$$

Table I: Details of Datasets

Datasets	***Number of apps***	***Number of benign apps***	***Number of malware apps***	***Number of features***
Malgenome -215	3,799	2,539	1,260	215
Drebin-215	15,036	9,476	5,560	215

2. *FPR (False Positive Rate):* The percentage of benign apps incorrectly classified (FN) to the total number of benign apps (TN+FP). FP (False Positive) is the number of incorrect predictions of benign and TN (True Negative) is the number of correct predictions of benign samples.

$$FPR = \frac{FP}{TN + FP}$$

3. *Precision:* The proportion of the correctly identified benign apps to all the predicted benign apps.

$$Precision = \frac{TP}{TP + FP}$$

4. $F_1\ score$*:* The harmonic mean of Precision and Recall. $F_1\ score$ is a better performance metric than the accuracy metric for imbalanced data [10].

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

The F-beta score is the weighted harmonic mean of precision of recall where F-beta value at 1 means perfect score (perfect precision and recall) and 0 is worst.

$$F_\beta = (1 + \beta^2)\frac{Precision \times Recall}{(\beta^2 \times Precision) + Recall}$$

When $\beta = 1$, F-beta is $F_1\ score$. The β parameter determines the weight of precision and recall. $\beta < 1$ can be picked, if we want to give more weight to precision, while $\beta > 1$ values give more weight to recall. Since we want to identify maximum number of malware apps, we give more weights to recall and utilize $\beta > 1$ values. Hence, the F-beta score is considered the principal performance metric to evaluate models in our experiments.

5. *Wilcoxon rank-sum test:* Wilcoxon rank-sum test evaluates the statistical significance of the model performance. The test checks the null hypothesis that two measurement sets are taken from the same distribution while the alternative hypothesis is that measurements are more likely to be higher in one study than those in the other.

C. Experimental Design

We evaluated our model performance by comparing it with the performance of LR, SVM, and DT methods. Both datasets were randomly split into training and test data while maintaining the apps class ratio between benign and malware samples. Trained data was used to train each of the models we experimented with while test data was used for evaluating the performance of the models. To verify the consistency of the model, we experimented with each of the models with 10-fold cross-validation.

The SVM, LR, and DT classifiers were applied to both datasets for comparing results with our proposed Droid-NNet. The algorithms were implemented using Python scikit-learn library with available hyperparameter options. 'rbf' (Radial Basis Kernel) were chosen for SVM, 'gini' index was chosen for DT, and L2 penalty was chosen for LR classifier.

Our proposed method is a deep neural network. We used 'ReLu' activation function in the hidden layer and 'sigmoid' function in the output layer. 'Adam' and 'binary cross-entropy' were used for optimizer and loss function respectively. We implemented an early stopping method to stop training once the model performance stops improving on the test data. We selected validation loss to be monitored for early stopping and set minimum delta to $1e-4$ (checks minimum change in the monitored quantity to qualify as an improvement) and patience to 10 (checks number of epochs that produced the monitored quantity with no improvement after which training will be stopped). Mini-batch gradient descent was considered and a batch size of 64 was chosen to train the model. The initial learning rate was set to 0.001 with a decay of $1e-5$ in every epoch. The L^2 regularization technique was applied to the output of the hidden layer to prevent the network from overfitting and the regularization parameter 'lambda' was set to 0.001. The 'beta' parameter in calculating the F-beta score was set to 10 to give more weight to recall so that the maximum number of malware apps can be identified. All the parameters and hyperparameters used in the model were optimized by grid search.

The experiments are carried out on a Windows 10 Intel(R) Core (TM) i7-8565U CPU 1.80 GHz with 16.0 GB RAM and NVIDIA GeForce MX250 2GB GDDR5. We implemented our experiment on Keras framework in Python 3.7 version.

D. Experimental results

We compared the results of Droid-NNet with the other three classifiers. The F-beta score was used to evaluate the models' performance. 10-fold cross-validation was performed for each of the experiments. The same configuration was applied to each Malgenome-215 and Drebin-215 datasets for maintaining consistency.

1) Performance evaluation on Malgenome-215 dataset: We trained our proposed network Droid-NNet for 100 epochs with early stopping. All the classifiers were trained on 90% of data and tested on the remaining 10% of the data. Table II illustrates the experimental results of different methods on this dataset. Droid-NNet outperformed other methods and achieved the highest F-beta score of 0.988157±0.007, whereas the second-highest was achieved by LR with 0.988859±0.006 F-beta score. The findings of FPR and TPR for Droid-NNet are also higher than other approaches. Table III presents the statistical significance of the performance of the models. Droid-NNet demonstrates a statistically better F-beta score than SVM, and DT, though superiority of Droid-NNet over LR was not statistically significant considering 0.05 significance level. The boxplot in Fig. 4 shows the distribution of F-beta score of ten folds.

Table II: Experimental results of different classifiers on Malgenome-215 dataset

Classifiers	***F-beta***	***TPR***	***FPR***
DT	0.978411	0.973810	0.019305
SVM	0.981564	0.961111	0.008280
LR	0.988859	0.976190	**0.004850**
Droid-NNet	**0.992623**	**0.988095**	0.005124

Table III: Statistical significance of Droid-NNet on Malgenome-215 dataset

Classifiers	***Statistical significance***
Droid-NNet vs. DT	0.0031971*
Droid-NNet vs. SVM	0.00407199*
Droid-NNet vs. LR	0.0881054

**Statistical significance considering 0.05 significance level*

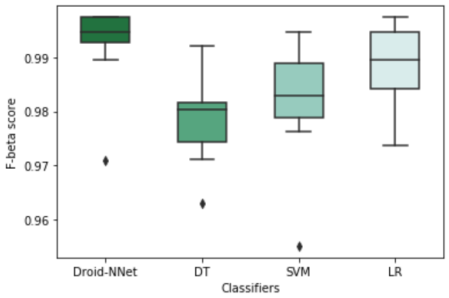

Fig. 4: Boxplot of F-beta score of ten folds on Malgenome-215 dataset

2) Performance evaluation on Drebin-215 dataset: The training and testing split ratio for all the classifiers was 90%: 10%. The experimental results of implementing our proposed network Droid-NNet and *the* other three algorithms on Drebin-215 dataset *are* illustrated *in* Table IV. Droid-NNet outperformed all the other three classifiers considering *the* F-beta score, FPR, and TPR. From Table IV, Droid-NNet had the maximum F-beta score (0.988157±0.002) whilst the nearest F-beta score was 0.978311±0.002 obtained by the DT classifier. Fig. 5 presents the F-beta score achieved in each of the tenfold by all the classifiers. The boxplot shows that Droid-NNet performed better than other classifiers with better consistency. Table V illustrates that the Droid-NNet provides a statistically significan*t* higher F-beta score than the other three classifiers considering 0.05 significance level.

Table IV: Experimental results of different classifiers on Drebin-215 dataset

Classifiers	***F-beta***	***TPR***	***FPR***
DT	0.978311	0.973176	0.018679
SVM	0.972919	0.953373	0.015619
LR	0.977644	0.962733	0.013613
Droid-NNet	**0.988157**	**0.979297**	**0.006648**

Table V: Statistical significance of Droid-NNet on Drebin-215 dataset

Classifiers	***Statistical significance***
Droid-NNet vs. DT	0.000157052*
Droid-NNet vs. SVM	0.000157052*
Droid-NNet vs. LR	0.000506541*

**Statistical significance considering 0.05 significance level*

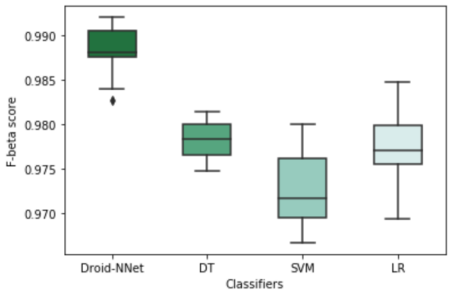

Fig. 5: Boxplot of F-beta score of ten folds on Drebin-215 dataset

V. CONCLUSION

Malware is increasingly posing a serious security threat to Android OS smart device users. It is essential to develop an automatic malware detection solution to reduce the risks of malicious activities. In this paper, we proposed a neural network-based framework, Droid-NNet, for Android malware detection. We train the Droid-NNet with L^2 regularization technique, early stopping criteria and mini-batch gradient descent method. We performed all the experiments on two datasets (Malgenome-215 & Drebin-215) of Android apps to evaluate Droid-NNet. We evaluated Droid-NNet performance by comparing it with the performance of LR, SVM, and DT algorithms. The experimental results show that Droid-NNet outperformed other methods and achieved the highest F-beta score for both the dataset.

REFERENCES

[1] Kakavand, Mohsen & Dabbagh, Mohammad & Dehghantanha, Ali. (2018). Application of Machine Learning Algorithms for Android Malware Detection. 32-36. 10.1145/3293475.3293489.

[2] Kalash, M., Rochan, M., Mohammed, N., Bruce, N. D., Wang, Y., & Iqbal, F. (2018, February). Malware classification with deep convolutional neural networks. In *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)* (pp. 1-5). IEEE.

[3] Mujumdar, A., Masiwal, G., & Meshram, D. B. (2013). Analysis of signature-based and behavior-based anti-malware approaches. *International Journal of Advanced Research in Computer Engineering and Technology (IJARCET)*, *2*(6).

[4] Burguera, I., Zurutuza, U., & Nadjm-Tehrani, S. (2011, October). Crowdroid: behavior-based malware detection system for Android. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices* (pp. 15-26). ACM.

[5] Gavrilut, D., Cimpoesu, M., Anton, D. and Ciortuz, L. (2009) Malware Detection Using Machine Learning, Proceedings of the International Multiconference on Computer Science and Information Technology , 735-741.

[6] Afrin, R., Haddad, H., & Shahriar, H. (2019, July). Supervised and Unsupervised-Based Analytics of Intensive Care Unit Data. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)* (Vol. 2, pp. 417-422). IEEE.

[7] Liu, Z., Zeng, Y., Yan, Y., Zhang, P., & Wang, Y. (2017). Machine Learning for Analyzing Malware. *Journal of Cyber Security and Mobility*, *6*(3), 227-244.

[8] Saracino, A., Sgandurra, D., Dini, G., & Martinelli, F. (2016). Madam: Effective and efficient behavior-based Android malware detection and prevention. *IEEE Transactions on Dependable and Secure Computing*, *15*(1), 83-97.

[9] Yu, W., Zhang, H., Ge, L., & Hardy, R. (2013, December). On behavior-based detection of malware on Android platform. In *2013 IEEE global communications conference (GLOBECOM)* (pp. 814-819). IEEE.

[10] Yerima, Suleiman Y., and Sakir Sezer. "Droidfusion: A novel multilevel classifier fusion approach for Android malware detection." *IEEE transactions on cybernetics* 49.2 (2018): 453-466.

[11] Y. Zhou and X. Jiang, "Dissecting Android malware: Characterization and evolution," in *Proc. IEEE Symp. Security Privacy (SP)*, San Francisco, CA, USA, May 2012, pp. 95–109.

[12] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, "Drebin: Efficient and explainable detection of Android malware in your pocket," in *Proc. 20th Annu. Netw. Distrib. Syst. Security Symp. (NDSS)*, San Diego, CA, USA, Feb. 2014, pp. 1–15.

[13] Ham, H. S., Kim, H. H., Kim, M. S., & Choi, M. J. (2014). Linear SVM-based Android malware detection for reliable IoT services. *Journal of Applied Mathematics*, *2014*.

[14] Sewak, M., Sahay, S. K., & Rathore, H. (2018, June). Comparison of deep learning and the classical machine learning algorithm for the malware detection. In *2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)* (pp. 293-296). IEEE.

[15] Saracino, A., Sgandurra, D., Dini, G., & Martinelli, F. (2016). Madam: Effective and efficient behavior-based Android malware detection and prevention. *IEEE Transactions on Dependable and Secure Computing*, *15*(1), 83-97.

[16] Duc, N. V., & Giang, P. T. (2018, December). NADM: Neural Network for Android Detection Malware. In *Proceedings of the Ninth International Symposium on Information and Communication Technology* (pp. 449-455). ACM.

[17] Alauthaman, M., Aslam, N., Zhang, L., Alasem, R., & Hossain, M. A. (2018). A P2P Botnet detection scheme based on decision tree and adaptive multilayer neural networks. *Neural Computing and Applications*, *29*(11), 991-1004.