Static Malware Detection

1st Ahmed Tamer
Dep of Computer Scince
Ahram Canadian University
Giza, Egypt
41810217.Ahmed@acu.edu.eg

2nd Yousef Hisham
Dep of Computer Scince
Ahram Canadian University
Giza, Egypt
41810305@acu.edu.com

line 1: 3rd Zeyad Mohamed
Dep of Computer Scince
Ahram Canadian University
Giza, Egypt
41810101.ziad@acu.edu.egef

4th Mohamed Ibrahim
Dep of Computer Scince
Ahram Canadian University
Giza, Egypt
41810273.mohamed@acu.edu.eg

5th Mostafa Abdelaziz
Dep of Computer Scince
Ahram Canadian University
Giza, Egypt
41810227.mostafa@acu.edu.eg

6th Ahmed Yousef
Dep of Computer Scince
Ahram Canadian University
Giza, Egypt
41810093.ahmed@acu.edu.eg

***Abstract*—We could achieve valuable results in terms of computational power and RAMs optimization, in the project of malware detection of the Canadian Center of Cyber Security. First of all, the dataset is enormously huge, and the task of reading the data looks like a problem to low computational power PCs, in the Canadian Center of Cyber Security approach, the way of reading the data doesn't save or optimize any computational power at all. Therefore, we formed a way in which we can read this vast data in our normal PCs is by reading the data as data frames, as "int" and reducing the int types to save more RAMs so we could read all of the data in a limited power PCs, not huge servers. Feature selection: it wasn't an option to read the whole data as a one-shot, to be able to perform our feature selection approach in one time. Neural Network Models: ANN could reach 86.3% accuracy, Radial Basis neural network 89.97% accuracy, Multilayer perceptron which got 90% accuracy, then CNN which is the most complicated since the dataset isn't images, which lead to use the complicated function for data image generation to enable the CNN to work on it but of course, this means more resources needed to generate images for all of this data, CNN got 91.5% accuracy. Machine learning models: ensemble model, random forest, logistic regression, decision tree, and KNN in which got 92% accuracy.**

I. Introduction

Android apps have reached 2,797,581 apps which are officially on the play store [1]. This is excluding apps that are published on other app stores. These big numbers of apps make traditional scanning whether by human inspection or traditional scans almost impossible. This has been the main motivation for the use of machine learning algorithms to analyze the pattern of malicious apps and detect it. Canadian Institute for Cybersecurity [2] has been hosting competition every year for the best datasets for malicious and benign apps. Also, the winners are the teams with the best datasets and the best predictive machine learning or deep learning models. Researchers have used different sets of features and analysis techniques (static, dynamic, and hybrid) to analyze and mitigate malware samples. The static analysis is all about detecting the android app according to the permissions the app requires to run. Dynamic analysis techniques are all about running the app in a virtual environment and detecting the way the malicious apps behave and benign apps behave. This research paper is more concerned about the static analysis which was implemented in this paper [3]. The research team was looking to improve their solution on the dataset they used. The dataset used was mainly generated by their researchers and was their biggest contribution to the problem as they have mentioned in the research papers. The dataset is a huge, labeled dataset containing 400,000 androids (200K malware and 200K benign) samples. The data is a representation of the permissions the apps require to run encoded in numbers and represented as features. The previous research paper proposed a CNN model [4]. This paper will go through their approach to improve the technical solution for the problem throughout the whole procedure.

II. Details of experiment

III. *we are using the dataset of Canadian Center for Cyber Security which has fourteen different malware types. First of all, Feature extraction from the APK file, there were 9504 features extracted in the first part of the experiment, then by using random forest classifier algorithm, 2465 best features were selected to feed to the model which is our second step (feature selection).*

IV. Generation of Data

Firstly, how does the data get generated, android applications end up in an android package kit called APK, which is basically a zip archive of AndroidMainfest.xml, we need to reverse engineer those apk files to extract the features (resources, accessibilities, etc.….) .

A. Nature of the data:

In the meantime, our dataset is simply consisting of 14 concatenated malware csv files and 5 non-malware files, it contains 15 different classes, fourteen classes representing fourteen different kinds of malwares (Adware, Backdoor, FileInfector, NoCategory, Potentially Unwanted Apps (PUA), Ransomware, Riskware, Scareware, Trojan, Trojan-Banker, Trojan-Dropper, TrojanSMS, Trojan-Spy and Zero-Day) and one non malware class.

Our dataset originally entails 9504 features.

B. Benign apps:

Androzoo dataset is used which now includes more than eight million distinctive android apps and the number is still growing, this dataset is collected from different resources such as official android market, Google Play, Anshi, AppChina, 1mobile, and Genome project dataset

C.Feature selection:

Feature selection is carried out by random forest classifier, at the beginning during the creation of the

forest, there is a value calculated for each feature called Gini importance which is a normalized total reduction, based on this value, we get to know the most important features, all the features are arranged in descending order according to its Gini importance and best k features are selected and the least important features are then dropped

V. MACHINE AND DEEP LEARNING METHODS

We have used auto-encoders after feature selection to ensure the integrity of feature selection. CNN was used and got a fairly good accuracy. RNN was used but it was complex for the data training and due to the fact of resulting vanishing gradient descend so we tried other variation of it which was LSTM. The LSTM didn't work well in our situation as it required very high computational power and it wasn't available for us at this time. We proposed machine learning models(Logistic regression classifier, Decision Tree Classifier, K-Neighbors Classifier and Random Forest classifier).

Model	***CNN***	***MLP***	***Ensemble***	***RNN/LSTM***	***ANN***	***RBNN***
Result	91.5	90	92.5	------	86.3	89.97

It is clear from the previously stated accuracies that the use of deep learning wasn't required as machine learning algorithms nearly got the same accuracy or in our approach it has got even higher accuracies.

A. Body

Starting with the dataset the main challenge is that the dataset is huge, it is a 7.5 GB of data at which when it's loaded by pandas "data frames" [1]. Our data consists of 384704 rows and 1238 that is equivalent to 476,267,266 numbers and in the previous solution, the researchers read it by default which represents 64 bits for each number. The data was read in the solution of the other researchers into the data frames in a total size of 28.5 GB ram. In our approach, we managed to read the 10000 records in the size of 90.5 MB instead of 750 MB so that makes our approach less exhausting for the ram as the total rams required are 3.56 GB of ram, we did this by specifying the data type of the data frame numbers to be only unsigned 8 bits. Reading all of the data at once is essential as there is a feature selection function that requires all of the data to be present in the ram to run either this or it will result in inconsistency of the data. Our second improvement was on the selected features. In the previous work, they have used the feature selection function in a way that they didn't make clear but got 2400 features, In our work, we run their feature selection function but only got 1238 features which resulted in nearly the same accuracy anyway. This could be considered another improvement as the data with nearly half the size got the same accuracy. We used auto-encoders in order to make sure that the 1238 features are actually the lowest features that we could get and that was proven,later we started using our machine learning and deep learning methods

B. Deep learning approach

1-Auto Encoder:

We used auto-encoders in order to make sure that the 1238 features are actually the lowest features that we could get and that was proven,later we started using our machine learning and deep learning methods..

2-CNN

2.1) Data Preprocessing

The first step was creating a function to convert the data into images because CNN is designed to process pixel data, a function called (Image_Creation) was created with two parameters X_Data_Best and Size_img. This function loops over x_data_best (which is the result of the feature selection function) row by row and then it removes the latest 224 feature to reshape it and convert it to a 2D array with a size of 32x32, then the 2D array is appended in a new initialized variable called (X_Data_Final), then we called the above function on (X_Data_Final), with the main data as its parameter to convert it to 2D arrays.

In the second step, we defined a function called (Label Encoding) with one parameter. Firstly a variable called (Y_Data_Final) was initialized and Y_Data was passed to it as its value then another variable called (le) was initialized and label_Encoder() which is already built-in was set as its value. Then we passed (Y_Data_Final) to le using "fit()" after that, we found the number of classes and converted it to a list. Then we replaced the old label with new labels and reshaped them to convert its shape to (384704, 1).

2.2) The Network

Two convolutional layers were used in this network in addition to one max-pooling layer of size 2x2.

2.3) Model

First, we import all the necessary modules required to train the model. The model was created using a batch size of 32 was used, it contributes massively to determining the learning parameters and affects the prediction accuracy. The network will be trained for 20 epochs.

Firstly, a convolutional layer was added because it was specifically designed for pixels, we used a Relu activation function to help the network learn what is non-linear, then a max-pooling layer was added. Finally, we add a dense layer with an activation function of "SoftMax" with a number of units equal to 15 which is the number of classes, because we are working on a multi-class classification problem.

Then an early-stopping was used to make training stops when the accuracy has stopped increasing.

After the model is created, it was compiled using the Adam optimizer, one of the most popular optimization algorithms. Then the loss type was specified which is categorical cross-entropy and used for multi-class classification. After that, we specified the metrics as accuracy.

2.4) Training

The last step is to train the model with Keras fit(). It will be trained for 20 epochs. The second step is storing the result of this function in history, to use it later to plot the accuracy and loss function plots between training and validation to analyze the model's performance visually.

3- LSTM

One of the models we used is LSTM (Long short term memory) in which it's a modified version of RNN (Recurrent neural networks). RNN is feedforward neural network that has an internal memory. RNN do the same function for every input of data while the output of the current input depends on the past one computation. After producing the output, it is copied and sent back into the recurrent network. For making a decision, it considers the current input and the output that it has learned from the previous input. But one of his disadvantages is gradient vanishing and exploding problems with big data and its training is very difficult task, to solve this problems we used LSTM which makes it easier to remember past data in memory. The vanishing gradient problem of RNN is resolved here. So we began to work with LSTM in the data but unfortunately we didn't get good accuracies in it because a lot of the challenges, some of this challenges that we used wrong numbers for trainable parameters in which it gives us under fitting and the training time, so we solve this parameters and put true numbers of trainable parameters but we can't run it because computational power problem so we stopped here with LSTM and began with another model which is ensemble model.

4-ANN: **Artificial Neural Network**

ANN uses the processing of the brain as a basis to develop algorithms that can be used to model complex patterns and prediction problems. ... In our brain, there are billions of cells called neurons, which processes information in the form of electric signals . in our ann we are using 20% of the data as testing , loss function of "categorical crossentropy" and optimizer "Adam" , we used three dense layers to give the output to their corresponding layers two of the three are using 'relu' as their activation function and the last one is using "sigmoid", it resulted in accuracy 86.3%.

5-Radial basis neural network

Radial basis function networks are distinguished from other neural networks due to their universal approximation and faster learning speed. An RBF network is a type of feed forward neural network composed of three layers, namely the input layer, the hidden layer and the output layer. 20% of the data as test, loss function "categorical crossentropy" and optimizer "Adam", one flatten layer , RBF layer then followed by dense layer with activation function sigmoid, it has accuracy of 89.97%.

VI. Machine learning approach:

Ensemble models is Building multiple models of different types and simple statistics are used to combine predictions and it's called voting ensemble which is one of the simplest ways of combining the predictions from multiple machine learning algorithms. We had done 2 ensemble models first model was with 3 different classifiers: Logistic regression classifier, Decision Tree Classifier and K-Neighbors Classifier, with this model we get 89% accuracy and the second model was with 4 different classifiers: Logistic regression classifier, Decision Tree Classifier, K-Neighbors Classifier and Random Forest classifier, with this model we get 92% accuracy

VI. Conclusion

for dimensionality reduction, we tend to use auto-encoders to make sure the Features selected are the most important features to get to run the model in the highest efficient way, but it turns out that auto-encoders function need unrealistic time to run on the whole data even we tried it on the 1238 feature because of the computational power, conclusively, we reached a fact that their wat of feature selection was correct and they picked the best features. All in all, Machine learning algorithms could get almost equal or even get higher accuracies than those of neural networks. Moreover, they need less computational power to work. obviously, machine learning algorithms is the way to go to work on these types of datasets.

References

[1] https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

[2] Uni of brunswick ,Android Malware Dataset (CIC-AndMal2017)

[3] DIDroid: Android Malware Classification and Characterization Using DeepImageLearning.Pages7082

[4] Keiron Teilo O'Shea, An Introduction to Convolutional Neural Networks.

[5] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

Recommendation

We had a lot of challenges in this project, the main part of it was computational power and limited RAM storage. Also one of the main considerations that we have is running the auto-encoders on the original data in order to get the most effective features in a more elegant way. Running LSTM on the data might render helpful results in the future but machine learning algorithms should be the main focus in terms of training for any future work for the model