Analysis of Malware Detection using Machine Learning

[1]Mohanraj A, [2]Ashrey Deepak Mudliar, [3]Gopi K and [4]Kavin Kumar V
Department of Computer Science and Engineering,
Sri Eshwar College of Engineering, Coimbatore, Tamil Nadu, India.
[1]mohanckpc@gmail.com, [2]ashrey.d@sece.ac.in.,[3]Nadu gopi.k@sece.ac.in,[4]kavinkumar.v2019@sece.ac.in.

Article Info
S. Venkatesh et al. (eds.), *Emerging Trends in Mechanical Sciences for Sustainable Technologies,* Advances in Intelligent Systems and Technologies,
Doi: https://doi.org/10.53759/aist/978-9914-9946-4-3_28

Abstract – Malware is simply the mechanism that performs malicious actions. It functions as an executable machine performed model which performs detection. Attackers always use the malware to steal the sensitive information. It is injected into the system and renders the whole component which then make the system to be not compatible to operate in the organization which in change a threat to many. It is broadly referred to many ofthe malicious malware substances like Worm, Trojan, Backdoor, Botnet, Ransomware, Rootkit.

Keywords – Malware, Android, Machine Learning, APK.

I. INTRODUCTION

Mobiles are a necessity in most people's daily lives, and the majority of phones currently use Android, with its sales accounting for a major segment of market shares for the past few years. With rise in the usage of Android, the increase in malwares targeting Android smartphones hasskyrocketed . Given the variety of Android programs (also known as apps), it is quiet impractical to devote human resources on assessing a subset of the provided application. (Since 2017 there have been more than millions of individual apps within the Google Playstore, with these numbers rapidly increasing.) Malware detection is purely based on software categorization. The two major processes include that of assessing and categorizing software. Firstly, with the usage of analysis multiple definations of a program can be derived, thence the characteristics are recognized from them. Classifiers then build classifications based on these attributes. As there is a large quantity of applications for evaluation, automated software analyzing and classification becomes a necessity. Neither of these procedures require human efforts.

Malwares specific to phones are an increasing issue in daily life.The indentification of malware is majorly just a program classification issue dependingon data collected from [1] The escalating threat indicates that the problem has not been resolved. It is quiet difficult to achieve fewer false positive rates, and with the access to a great number of labelled malware andexamples are required for a start to obtain reasonableprecision. The majority of previous research has been accomplished on malware sample collections that are relatively small [2, 3]. A successful result for both procedures critically depends on the measures used for computing the difference between an image and the other or class means. The machine learning and computer vision communities fields have paid immense attention in metric learning and has significantly imporved thefunctioning of distance based classification. As aresult, we have framed our classification learning problem as discovering a metric that is shared by all platforms. We employ the Large Margin Nearest Neighbor (LMNN) framework for k-NN classification [4]

Inter-class transfer, on the other hand, has no higher goal.speed, but with better generalisation performance, in most cases for object classes with a limited number of training instances prior distributions can be deduced from known object classes.in terms of distortions, over the expected intra-classvariance[5] The malicious payload app would not be detected by the existing systems if it was included in the static code.Obfuscation, in addition to native code, is a problem with malware detection on Android. Even professional developers are required to secure their Android applications to protect them from reverse engineering attacks.[6] The researchers used a random forest on a dataset of linked smart-phones to detect fraudulent code .They ran APKs through an emulator and noted several aspects suchas permission, memory information, and network traffic .For categorization, as well as evaluating the approach in various tree sizes[11] A system proposed by Moser et al. for understading numerous execution ways by providing variable inputs and scenerios to simulate working of similar malwares[7, 8] Further harnessed by Scikit for providing state- of-the-art representation of various popular machine learning algorithms, with a well maintained easy to use platform that is used the Python programming language.[9, 10]

Traditional sequence classification algorithms frequently convert a sequential characteristic like symbol count

n-grams. Aside from individual API call frequencies, the count of combinations of 1 or 2 calls, and so on. The bag-of-words representation, however, loses sequential data. Although it preserves local order information, the n-gram representation does not have flexibility. When n becomes somewhat larger, most n grams appear just once, in one of the sequences.This significantly reduces the generalisation power of classifiers that use this sort of feature. We look at the efficacy of Deep Neural Networks (DNNs) for call based API software categorization. The self learning capacity of characteristics is critical for software categorization, particularly in case ofmalware. A method that needs handmadequalities is sluggish and hence incapable of responding quickly to malware outbreaks. DNNs may also generate sophisticated and flexible features that aid in classification generalization. For sequence classification, we created a Convolutional Neural Network (CNN) that performs convolution operations down the line, understanding the patterns in every site as the convolution glides along the line. In addition, the design combines many layers to build top features from tiny characteristics.

II. RELATEDWORK

Convolutional Neural Networks (CNNs) have been used extensively in image recognition research during the last several years. The renowned picture classification task, ImageNet 2010, was won for the first time by a deep learning system called AlexNet. Since then, deep neural networks have won competition, as contrast to prior strategies, which concentrated on combining an ensemble of hand-generated machine learning algorithms. Other significant recent ImageNet competition outcomes include GoogleNet in 2014 and the usage of Resnets in 2015. All of these networks contain a convolutional element at their core, clearly proving the spatial locality emphasis inherent in CNN is ideally matched with the task of picture classification. [11-12]

To make a final classification, most malware detection programmes combine handcrafted characteristics such as SVM. The two Android malware analyzersAndrosimilar and DenDroid are two programs that generate signatures for code blocks further utilising these signatures to classify malware using SVM and other clustering methods. There are various technologies focusing on categorization of malwares by studying Call Flow Graphs The key advantage of the method over conventional approaches is that,DNNs process characteristics directly from data, eliminating the requirement of manual efforts . Therefore, DNN features are highly adaptable and generalized than the ones produced from basic algorithms. One fault of DNN high demand.Relatively very little researches are published on the use of deep learning algorithms for identifying malware. The published title provides a focus on employing a relative minute density of linked networks to classify APK characteristics. DroidDetector

III. MODULE DECSRIPTION

Android App

An APK file archive is used to bundle the required files of the application. The following terms are found in the APK such as: res folder, assets folder, classes.dex file. The classes.dex and the Andriodmanifest.xml are one of the important segments of the work. The res and the assets folder contains the various necessary files to run the application and also picture and sound files. The bytecode and the major sources of runnable code are contained in the classes.dex file although there are alternative loading codebases within the APK. Despite the recent increase in malwares being launched from native code libraries, a majority of malwares utilise the dex code.In case of our analysis, the open source tool Androguard is used to obtain the code generation of classses.dex

DataSet

```
import pandas as pd
import numpy as np
np.random.seed(0)
from sklearn.metrics import precision_score,r ecall_score,f1_score
import tensorflow as tf
tf.compat.v1.set_random_seed(0)
from tensorflow import keras import matplotlib.pyplotmpt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncode r
from sklearn.metrics import ConfusionMatrixDi splay
from sklearn.metrics import confusion_matrix
data = pd.read_csv("../input/android-malwar e-dataset-for- machine-learning/drebin-215-d ataset-5560malware-
9476- benign.csv")
print("Total missing sum : ",sum(list(da ta.isna().sum())))
data
print("Total Features : ",len(data.columns)-1)
mpt.bar(classes,count)
mpt.title("Class balance")
mpt.xlabel("Classes")
mpt.ylabel("Count") mpt.show()
```

```
mpt.figure(fsize=(10,3)) mpt.subplot(3,1,4)
mpt.plot([str(i) for i in range(1,ep+1)],hist
ory.history['accuracy'],label="Train Accuracy ")
mpt.plot([str(i) for i in range(1,ep+1)],
hist ory.history['val_accuracy'],label="Validation Accuracy")
mpt.legend() mpt.xlabel("Epoch") mpt.ylabel("Accuracy")
mpt.title("Epoch vs Train Loss")
mpt.subplot(1,2,2)
mpt.plot([str(i) for i in range(1,ep+1)],hist
ory.history['loss'],label="Train Loss") mpt.plot([str(i) for i in
range(1,ep+1)],hist ory.history['val_loss'],label="Validation Los s")
mpt.legend() mpt.xlabel("Epoch") mpt.ylabel("Loss")
mpt.title("Epoch vs Validation loss")
mpt.show() //Fig. 2 It will print confusion matrix.
ypred = model.predict(test_x) for x in range(len(ypred)):
 if ypred[x] > (1-ypred[x]): ypred[x]=1
else: ypred[x]=0
print("Precision : ",precision_score(testy,ypred)*100)
print("Recall : ",recall_score(testy,ypred)*100)
print("F1 Score : ",f1_score(testy,ypred)*1 00)
```

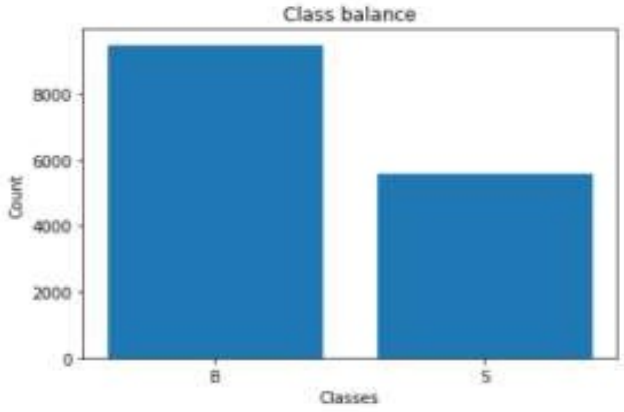

Fig 1. Class Balance.

Fig 1 shows the class balance.

IV. EXPERIMENT AND RESULT ANALYSIS

This study focuses on programme analysis that looks at calls an app makes to the Android system API. API calls outline the communication with the OS that powers it. These communications are necessary for an app's task completion, therefore they provide important details about an app's behaviours. An analyser for pseudo- dynamic programmes was created and put into use. Every time the analyser runs, it monitors a potential course of programme execution and creates a series of API calls along the way. Each programme becomes a collection of these sequences after several iterations of the analyser **Fig 2** shows the confusion matrix.

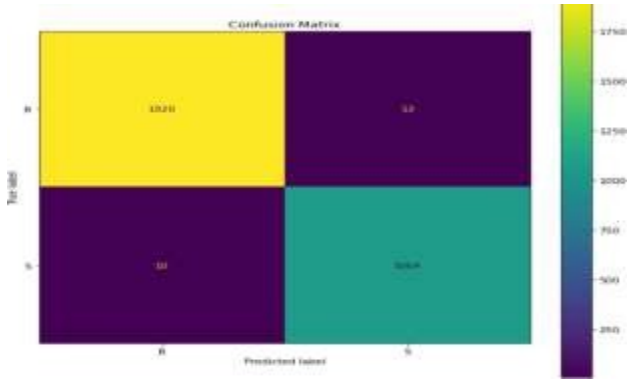

Fig 2. Confusion Matrix.

We examine Deep Neural Networks' (DNNs') utility in classifying software based on API calls. The capacityof the DNN to automatically learn characteristics crucial. A method that calls for handmade features is inherently slow and cannot respond quickly to malware epidemics. Additionally, complex and adaptable features produced by DNNs can aid in classification generalization.

V. CONCLUSION

The application of deep learning over MLAs have given better results. As the time of training is proportional to number of layers an increase in the same is not quiet functional.This thence increases the test time complexity. While our study established the efficiency of DNNs for Android app categorization, there is still much potentialfor improvement. We concentrated on API-call sequences from an application's core code in particular. Many more pieces of information may be extracted from a programme. For a future scope the creating of deep neural network that is capable of taking various channels from APKs that could be combined with data to provide impoved categorization.

References

[1]. Statista, https://www.statista.com/.

[2]. M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, “Data mining methods for detection of new malicious executables,” Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001, doi: 10.1109/secpri.2001.924286.

[3]. J.Zico Kolter and Marcus A. Maloof, “Learning to detect and classify malicious executables in the wild,” in Journal of Machine Learning Research, pp. 2721–2744, 2006.

[4]. Weinberger, K., Saul, L, “Distance metric learning for large margin nearest neighbor classification”,Journal of Machine Learning Research 10, 207–244, 2009.

[5]. M. G. Miller, N. E. Matsakis, and P. A. Viola, “Learning from one example through shared densities on transforms,” Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662), doi: 10.1109/cvpr.2000.855856.

[6]. S. Alam, Z. Qu, R. Riley, Y. Chen, and V. Rastogi, “Droidnative: Semantic-based detection of android native code malware,” arXiv preprint arXiv:1602.04693, 2016.

[7]. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, pp. 1097–1105, 2012.

[8]. Moser, C. Kruegel, and E. Kirda, “Exploring Multiple Execution Paths for Malware Analysis,” 2007 IEEE Symposium on Security and Privacy (SP ’07), May 2007, doi: 10.1109/sp.2007.17.

[9]. Z. Yuan, Y. Lu, and Y. Xue, “Droiddetector: android malware char- acterization and detection using deep learning,” Tsinghua Science and Technology, vol. 21, no. 1, pp. 114–123, 2016.

[10]. Desnos et al., “Androguard-reverse engineering, malware and goodware analysis.\

[11]. H. Anandakumar, R. Arulmurugan, “Early Detection of Lung Cancer using Wavelet using Neural Networks Classifier”, Book Title “Computational Vision and Bio Inspired Computing” Lecture Notes in Computational Vision and Biomechanics, Springer Book Series, Volume No – 28, Chapter No – 09, ISBN 978-3-319-71766-1.

[12]. Niranjani and N. S. Selvam, “Overview on Deep Neural Networks: Architecture, Application and Rising Analysis Trends,” EAI/Springer Innovations in Communication and Computing, pp. 271–278, 2020, doi: 10.1007/978-3-030-44407-5_18