Grenze International Journal of Engineering and Technology, June Issue

Image based Android Malware Detection System using ResNet50 and VGG16

Rajeshwari Gundla[1] and Sachin R Gengaje[2]
[1]Department of Computer Science & Engineering, Walchand Institute of Technology, Solapur, India
Email: radhikagundla22@gmail.com
[2]Department of Electronics and Computer Engineering, Walchand Institute of Technology, Solapur, India
Email: srgengaje@witsolapur.org

***Abstract*— Attacks on mobile devices, such as smartphones and tablets, have increased as a result of their rising popularity. One of the most significant risks is mobile malware, which may lead to both security breaches and financial losses. Mobile malware will probably continue to develop and spread since it is used to commit different types of cybercrimes on mobile devices. As the Android operating system has gained popularity, mobile malware targets it especially. Users' security is seriously jeopardised by the quick spread of Android malware apps, which also makes it challenging to manually and statically analyse harmful files. Therefore, it is essential to accurately identify and categorise Android harmful files. There has been some discussion of Convolutional Neural Network (CNN) based approaches in this area, however there is still opportunity for performance enhancement. In this work, using two well-known deep learning models, ResNet-50 and VGG16, we present a transfer learning strategy to effectively detect the Android malware files. By converting malicious and benign APK files into grayscale images, the proposed model is trained on the AndroHealthCheck dataset. On the AndroHealthCheck dataset, our model outperforms state-of-the-art works in terms of accuracy, recall, precision, and F1 measures.**

***Index Terms*— Android Malware, Convolutional Neural Network, Deep Learning, ResNet50, VGG16.**

I. INTRODUCTION

A programme that has malicious intent is referred to as malware (malicious software). They are made to obstruct regular device operation, show unwanted advertising, steal private information, or remotely take over the user's device. Malware comes in many different forms, including Trojan horses, worms, ransomware, rootkits, and botnets [1]. Over time, malware has developed to become more complex. Malware employs polymorphic and metamorphic strategies to avoid being discovered by traditional malware detection methods [2–4].

The security of the Android ecosystem was seriously threatened by the emergence of new malware families and the rise in the number of mobile malwares. By conducting numerous studies relating to the detection and classification of Android malware samples, some studies shed light on how to address the security challenges faced by the Android ecosystem [5,6]. Despite the advancements made in the detection and classification of Android malware, a new problem has emerged because malware's destructive behaviour has changed over time. This makes it a difficult task for malware classifiers relying on machine learning to prevent performance deteriorations.

The methods that are thought to be most crucial for identifying malicious software are static analysis and dynamic

Grenze ID: 01.GIJET.10.2.80

analysis. Malware can be located using a combination of behavioural and signature techniques. Without running the programme, static analysis can be performed when a portion of source code appears suspect. To extract features, the source code must be deconstructed [7,8]. This method is not resistant to loading dynamic code or cryptic code [9–12]. The dynamic analysis, on the other hand, looks at the traits and signs of possible application during implementation [13,14]. Although dynamic analysis is a promising method, it requires a lot of time and resources [9,15]. Dynamic analysis and anti-emulation technology are used by intelligent malware [16–18].
Additionally, using static and dynamic approaches on such files requires a significant amount of manual work or human involvement. This requires domain expertise, to analyse or reverse-engineer the application [19].
The prevalence of attacks against the Android mobile operating system has drawn a lot of interest in the academic and business areas to the detection of Android malware. But the number of malicious programmes that are uploaded every day and the quantity of work that has been done are significantly different.
Deep learning and optimisation techniques have recently been tested in an effort to decrease the frequency of malware infections. The proposed work converts Android benign and malware applications to greyscale images. A CNN was developed to automatically extract the rich features of benign and malware greyscale images using a proven and widely accepted method for classifying images. As a result, these characteristics were used to classify benign and malicious applications.
The main contributions are listed as follows:

1. We propose a Framework consisting of ResNet-50 and VGG-16 for identification of malware and benign Android APK.
2. Malware and benign applications from dataset are converted to grayscale images.
3. These methods ResNet-50 and VGG-16 are implemented on PC using AndroHealthCheck dataset and their results have been analyzed. AndroHealthCheck dataset contains 215 features, 12300 malware and 4000 benign applications.

II. Literature Survey

The proposed deep neural network-based methods for detecting Android malware were summarised in this section. The following techniques are commonly used to categorise Android malware detection methods: (1) static analysis, which uses reverse engineering techniques to disassemble the app and then investigate app resources without actually executing the app. (2) dynamic analysis, which executes the app in an isolated and controlled environment like an Android Virtual Device (also known as an Android emulator) or a real device in order to monitor the app's behaviour during runtime.
Droid Malware Detector was proposed 1-dimensional CNN approach. Six convolutional layers and two dense layers made up their CNN model. One problem with this study is that, in contrast to the previous work, where Android malware datasets often contain APK (Android Application Package) archives, the dataset utilised to test the proposed model in this study comprises Microsoft Windows binaries [20].
A method based on long short-term memory (LSTM) was suggested by authors in [21]. The suggested model outperformed conventional machine learning techniques like kNN (k-Nearest Neighbours), SVM (Support Vector Machines), and AdaBoost, according to the experimental results. Additionally, the stacked LSTM model with three hidden layers offered superior accuracy than the conventional LSTM network with one hidden layer. The proposed model's highest level of accuracy was measured at 93.9%.
A hybrid deep learning model for Android malware detection called Deep Vis Droid is based on combining deep learning methods with image-based features. Grayscale images are extracted by Deep Vis Droid from the provided apk archives. The features dataset used to train and test their proposed deep neural network was then built using both local and global features descriptors. The results of the studies show that the Deep Vis Droid has an accuracy of 98.96% [22].
The study [23] examines the research on deep learning methods for malware detection. CNN, RNN, LSTM, and auto encoders are some of the deep learning techniques utilised for malware detection. It is discovered that LSTM has memory in the cell to have greater chances. In order to detect anomalies (malware), auto encoders are proven to have a better unsupervised approach with encoding and decoding. Deep learning and machine learning have made significant advances to the field of Android malware detection. This work offers information that motivates additional deep learning research, which is necessary to advance the state of the art.

III. Methodology

Transfer learning applies the already pre-trained model to a new problem with little to no modification. A machine improves its ability to generalise for a subsequent task by using the knowledge it obtained from a prior task. Instead

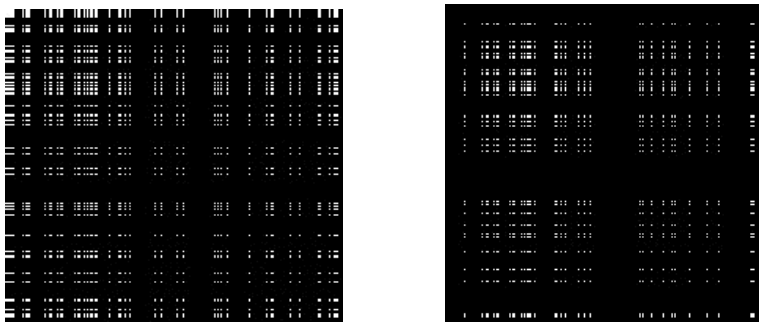
of starting from zero with training and fine-tuning, it enables us to construct a more durable architecture in the most economical manner. For transfer learning, a variety of models, including AlexNet, ResNet, and VGG, can be employed. They layered a lot of convolutional layers, which made it challenging to optimise the networks and caused deterioration and vanishing gradient issues. Various CNN models, which need end-to-end training, are used to detect Android malware. In this study, end-to-end training is substituted with the pre-trained ResNet-50 model. This speeds up the processing time and enhances the categorization outcomes. The commonly used architecture ResNet-50 and VGG-16 are useful for completing challenging jobs and enhancing detection performance.
The Android Malware Detection Dataset consists of different flavours and diversity of malware APK files that can be used for malware detection using machine learning. We used AndroHealthCheck dataset [24] in our proposed model. AndroHealthCheck dataset is in CSV format which consists of 0/1 values per cell for 215 features. This dataset is converted into grayscale image dataset. Algorithm for converting these set of features dataset to images is shown below:
An algorithm for creating images from existing datasets of malware and benign samples:
Algorithm: To create greyscale image dataset from set of features
Input: set of features $f_1, f_2, \ldots, f_{215}$
Output: set of images $img_1, img_2, \ldots, img_n$
for each android sample [1 … n] do

$$m= [f_1, f_2, \ldots, f_{215}]_{1 \times 215}$$

$j = [Transpose\_matrix(m)]_{215 \times 1}$
// we will use numpy libraries array broadcasting feature of matrix addition, it will convert m and j matrix to 215 X 215 size when we do matrix addition by repeating same features.

$$k = [[m]+[j]]_{215 \times 215}$$

Replace value in k matrix to 0 if value is not equal to 2;
Convert 2D array to image by assigning black color to 0 value cell and white color to 2 value cells;
Save image;
End
Fig.1 shows some of malware and benign greyscale images which are then used as input to pretrained models ResNet50 and VGG16.



a. Benign image b. Malicious image

Fig.1: Illustration of some of the benign and malware image

After converting to images, we apply VGG16 and Resnet50 on image dataset separately. Fig.2 shows flow diagram for image based android malware detection using ResNet50 and VGG16. Here pretrained models ResNet50 and VGG16 are used for android malware detection. These methods are implemented on PC using AndroHealthCheck dataset and their results have been analyzed. AndroHealthCheck dataset contains 215 features, 12300 malware and 4000 benign applications.

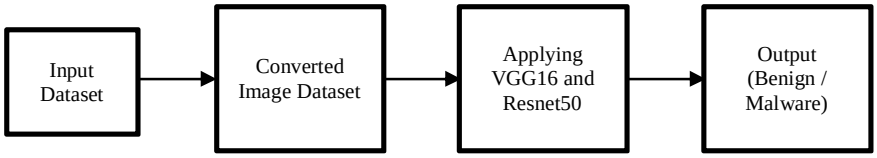


Fig. 2: Flow diagram of android malware detection using ResNet50 and VGG16

IV. Experimental Results

Performance Metrics

In this study, we employed an unbalanced set of data. As a result, we are unable to evaluate the performance of the models using the accuracy scale.
Following performance metrics are used in evaluation:

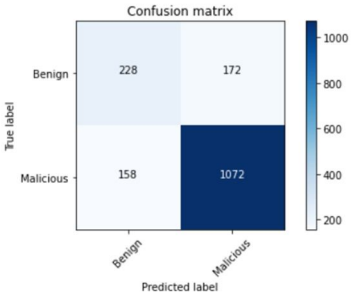
- *Precision* - Percentage of malicious apps identified that are actually correct = (TP) / (TP + FP).
- *Recall / True Positive Rate (TPR)* - Percentage of actual malicious apps that are correctly identified = (TP) / (TP + FN).
- *False Positive Rate (FPR)* - FPR is the ratio of negative samples that are wrongly classified as positive = (FP) / (FP + TN)
- *F1 Score* - Combines precision and recall into a single-number weighted average = (2 × Precision × Recall) / (Precision + Recall).
  Where
- *TP*: represents the number of Android applications as benign.
- *TN*: represents the number of Android applications as malware.
- *FP*: represents the number of benign Android applications as malware.
- *FN*: represents the number of malware Android applications as benign.

For unbalanced datasets [11], F-measure (F1 Score) is a better metric than other. So, in our experiments we used F-measure as the primary predictability indicator.
The evaluation metrics such as precision, recall and F1 score for image based android malware detection on dataset using ResNet50 and VGG16 are shown below:

Output: ResNet50

	precision	recall	f1-score	support
0	0.59	0.57	0.58	400
1	0.86	0.87	0.87	1230
accuracy			0.80	1630
macro avg	0.73	0.72	0.72	1630
weighted avg	0.80	0.80	0.80	1630

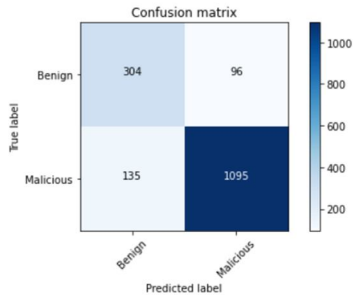
```
Confusion matrix, without normalization
[[ 228  172]
 [ 158 1072]]
```



Output: VGG16

	precision	recall	f1-score	support
0	0.69	0.76	0.72	400
1	0.92	0.89	0.90	1230
accuracy			0.86	1630
macro avg	0.81	0.83	0.81	1630
weighted avg	0.86	0.86	0.86	1630

```
Confusion matrix, without normalization
[[ 304   96]
 [ 135 1095]]
```



V. Conclusion

Smartphone users are increasingly at danger from Android malware. The creation of an effective Android malware detection system is essential to lowering the danger of malicious actions. In this study, we proposed a classification model for Android malware that uses ResNet-50 and VGG-16. The ResNet-50 and VGG-16 was used because it has transferable learning abilities. All experiments were conducted on the Andro Health Check dataset which contains 215 features, 12300 malware and 4000 benign applications in order to evaluate our model. Performance of our model is increased by VGG16 as compared with Resnet50.

References

[1] Qamar, Attia, Ahmad Karim, and Victor Chang. "Mobile malware attacks: Review, taxonomy & future directions." Future Generation Computer Systems 97 (2019): 887-909.

[2] S. Dong, M. Li,W. Diao, X. Liu, J. Liu et al., “Understanding android obfuscation techniques: A largescale investigation in the wild,” in Proc. of the Int. Conf. on Security and Privacy in Communication Systems, Singapore, pp. 172–192, 2018.

[3] G. Suarez-Tangil, “DroidSieve: Fast and accurate classification of obfuscated android malware,” in Proc. of the Seventh ACM on Conf. on Data and Application Security and Privacy, Scottsdale, AZ, USA, pp. 309–320, 2017.
[4] K. Bakour,H.M. ünver andR.A.Ghanem, “Deep camouflage: Evaluating android’s anti-malware systems robustness against hybridization of obfuscation techniques with injection attacks,”Arab Journal for Science and Engineering, vol. 44, pp. 9333–9347, 2019.
[5] S.Y.Yerima,M.K.Alzaylaee,A. Shajan and P.V., “Deep learning techniques for android botnet detection,” Electronics, vol. 10, no. 519, 2021.
[6] M. A. Albahar, M. S. ElSayed and A. Jurcut, “A Modified ResNeXt for Android Malware Identification and Classification,” Computational Intelligence and Neuroscience, Hindawi Limited, vol. 2022, pp. 1–20, 2022.
[7] A. Feizollah, N. B. Anuar, R. Salleh, G. Suarez-Tangil and S. Furnell, “AndroDialysis: Analysis of android intent effectiveness in malware detection,” Computers & Security, vol. 65, pp. 121–134, 2017.
[8] Martín, H. D. Menéndez and D. Camacho, “MOCDroid: Multi-objective evolutionary classifier for android malware detection,” Soft Computing, vol. 21, no. 24, Springer Science and Business Media LLC, pp. 7405–7415, 2016.
[9] W.Wang,M. Zhao, Z. Gao, G. Xu, H. Xian et al., “Constructing features for detecting android malicious applications: Issues, taxonomy and directions,” IEEE Access, vol. 7, pp. 67602–67631, 2019.
[10] A. Naway and Y. Li, “A review on the use of deep learning in android malware detection,” arXiv2018, arXiv:1812.10360, 2018.
[11] O. Aslan and R. Samet, “A comprehensive review on malware detection approaches,” IEEE Access, vol. 8, pp. 6249–6271, 2020.
[12] S. Venkatraman, M. Alazab and R. Vinayakumar, “A hybrid deep learning image-based analysis for effective malware detection,” Journal of Information Security and Applications, vol. 47, pp. 377–389, 2019.
[13] H. Cai,N. Meng, B. Ryder andD. Yao, “DroidCat: Effective androidmalware detection and categorization via app-level profiling,” IEEE Transactions on Information Forensics and Security, vol. 14, pp. 1455–1470, 2019.
[14] W. You, B. Liang,W. Shi, P.Wang and X. Zhang, “TaintMan: An ART-compatible dynamic taint analysis framework on unmodified and non-rooted android devices,” IEEE Transactions on Dependable and Secure Computing, vol. 17, pp. 209–222, 2017.
[15] A. Sadeghi, H. Bagheri, J. Garcia and S.Malek, “A taxonomy and qualitative comparison of program analysis techniques for security assessment of android software,” IEEE Transactions on Software Engineering, vol. 43, pp. 492–530, 2017.
[16] P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M. S. Gaur et al., “Android security: A survey of issues, malware penetration, and defenses,” IEEE Communications Surveys & Tutorials, vol. 17, pp. 998–1022, 2015.
[17] M. K. Alzaylaee, S. Y. Yerima and S. Sezer, “Emulator vs. real phone: Android malware detection using machine learning,” in Proc. of the 3rd ACM on Int.Workshop on Security and Privacy Analytics, Scottsdale, AZ, USA, pp. 65–72, 2017.
[18] T. Vidas and N. Christin, “Evading android runtime analysis via sandbox detection,” in Proc. of the 9th ACM Symp. on Information, Computer and Communications Security, Kyoto, Japan, pp. 447–458, 2014.
[19] F. Idrees, M. Rajarajan, M. Conti, T. M. Chen and Y. Rahulamathavan, “PIndroid: A novel android malware detection system using ensemble learning methods,” Computers & Security, vol. 68, pp. 36–46, 2017.
[20] Sharma, Arindam, Pasquale Malacaria, and M. H. R. Khouzani. "Malware detection using 1-dimensional convolutional neural networks." 2019 IEEE European symposium on security and privacy workshops (EuroS&PW). IEEE, 2019.
[21] Vinayakumar, R., et al. "Detecting Android malware using long short-term memory (LSTM)." Journal of Intelligent & Fuzzy Systems 34.3 (2018): 1277-1288.
[22] Bakour, Khaled, and Halil Murat Ünver. "DeepVisDroid: android malware detection by hybridizing image-based features with deep learning techniques." Neural Computing and Applications 33 (2021): 11499-11516.
[23] Majid, Al-Ani Mustafa, et al. "A review of artificial intelligence based malware detection using deep learning." Materials Today: Proceedings 80 (2023): 2678-2683.
[24] Dr. Prerna Agrawal, Dr. Bhushan Trivedi, July 20, 2021, "AndroHealthCheck Dataset", IEEE Dataport, doi: https://dx.doi.org/10.21227/gnsw-m336.