International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 08, August-2022

Detection of Smart Android Malware Employing Deep Learning and Machine Learning

Mrunmayi V. Kaware
MIT School of Engineering
MIT ADT University
Pune, India

Prof. Dr. Ayesha Butalia
MIT School of Engineering
MIT ADT University
Pune, India

***Abstract* - The bulk of smartphones now marketed have used the operating system Android. The open-source nature of the Android OS, leads to significant growth in Android malware. Malware can collect user information and can steal the privacy. As well as more sophisticated Artificial neural networks and deep convolution networks are two examples of machine learning techniques, more traditional machine learning strategies like Decision trees and K-Nearest Neighbor are investigated. The calculation of a decision tree can occasionally become rather complex when compared to different algorithms. The calculation of a decision tree can occasionally become rather complex in contrast to alternative algorithms. Decision trees often requires more time during the training phase. The study of machine learning-based techniques for identifying Android malware has grown recently. Artificial intelligence is increasingly embracing deep learning, a machine learning research area that is still very young. The technique of spotting Android malware has gradually grown in popularity due to the growing use of this in recent years. With this study, we suggest combining static and dynamic analytical characteristics of Android deep learning methods for malware detection applications. To determine if an app is dangerous or not, we suggest using TensorFlow, an Android malware detection algorithm, to provide a percentage-based accuracy. We utilise the 50000 data sample Malware Detection Dataset to carefully analyse the TensorFlow algorithm. The dataset used in this study includes thousands of Android apps, and it delves deeply into the features that deep learning uses to categorise malware. According to the study, deep learning is effective for recognising Android malware and it performs noticeably better when numerous training data sets are available. Keywords: Deep learning, malware, Android, detection, machine learning, trojan, worms.**

I. INTRODUCTION

The Android operating system has numerous advantages, including open source, scalability, and ease. As a result, it has become the world's most widely used intelligent interface operating system. However, due to Android's openness, many relatively basic apps on the market include a large quantity of malware. Malware on mobile devices has posed a significant threat to cyber security. While many Android apps are useful, personal privacy and sensitive information (such as payment account information and passwords) continuously at risk. A range of infiltrative or destructive computer code, such as Malware includes viruses, worms, rootkits, backdoors, spyware, Trojan horses. In essence, malware is malicious software or computer code. Malware is renowned for being specifically created to harm, interfere with, steal, or impose other unhealthy or illegal behaviours. Practically any Android information processing device that runs user applications is susceptible to malware infection and damage.

Also, it's propagation and defence mechanisms have been thoroughly researched for personal computers. The most recent methods for detecting malware on mobile platforms, especially on smart phone devices. Malware poses a serious risk to computer users' security and can cost businesses a lot of money. Depending on how they behave, malware detection has evolved into a dynamic challenge for researchers. The two primary types of malware detection and analysis techniques static and dynamic analysis, respectively. An executable file can be examined and its contents extracted via static analysis, which is done without actually running the file. The act of running malware and watching how it behaves on the system is known as dynamic analysis. When a new malware variant is found, specialists usually manually analyse the sample or create a tool that can compare the malware's similarities. This paper discusses malware detection using dynamic analysis. In dynamic analysis, we perform classification and measure the accuracy in percentage of the dataset of the application. Using the tensorflow algorithm, based on the accuracy level, we will detect the malware present in the android dataset. The author asserts that the article illustrates and emphasises a framework for active learning that is based on SVM, and that tests conducted inside the machinery validating the efficacy of an approach that demonstrates the targeted methodology's viability relevancy and measurability and can identify unknown malware [5].

II. RELATED WORK

A lot of books have been written about android malware detection techniques. We employ a variety of learning strategies for this. We provide the literature survey of work done in this field in this section. Discuss a few machines learning and artificial intelligence techniques that have been used to detect malware. Many approaches to machine learning are explored, including support vector machines, decision trees, and k-nearest neighbours [1]. A code behaviour signature- based malware detection system using a SVM rule to effectively detect malicious code and variations in real time and dynamically extend malware characteristics information was proposed in this paper. A high rate of detection and the strategy includes a low percentage of false positive and false negative results.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 08, August-2022

According to experimental data, On the first system, the effects of power and performance can even be disregarded. The method extracts a number of parameters and trains an offline support vector machine in order to take advantage of the greater computing capacity of a server or cluster of servers (off-device) [5].

A thorough analysis of Android malware detection via machine learning methods is provided in the study. We give a brief review of Android applications that covers the system architecture, security measures, and malware classification of the operating system. The author then examines the present status of research from important angles, with a focus on machine learning, such as sample collecting, data pre- processing, detection effectiveness assessment, feature selection, and machine learning models, algorithms. The research's outlook for machine learning-based malware detection for Android is then examined [6].

Five different types of features are originally gleaned by the author via static analysis of Android applications. Then, to learn features from Android apps, we develop a deep learning model. Therefore, unidentified Android malware is detected using the learned properties [7]. The paper asserts that deep learning techniques can totally avoid this stage. In real-time circumstances, many of them, according to new research, used a warped dataset that is entirely unusable. As a result, work is being done to create a new deep learning algorithm and architecture for malware detection. By using customised Convolution Neural Networks for detecting patterns in malware sequences that implement the weight sharing idea. By combining this with recurrent neural networks [8,] we may also detect recurrent malware patterns.

III. DATASET OF ANDROID APPLICATION

To compare the effectiveness of conventional a huge dataset, deep learning approaches, and machine learning pertinent data and a variety of Sample types is needed. Unfortunately, there isn't a publicly accessible dataset for study on malware and cybersecurity. The reason for this absence is the privacy-protecting policies. Given that every company has its own privacy policy, it is challenging to assemble a sizable dataset for virus detection. According to statistics, malware has increased in recent years, making it challenging to keep track of all of these spreading viruses and their cousins in a single place. Despite the fact that researchers are significantly advancing this field. their findings are still not considered because there isn't a single dataset that can contain all of the necessary samples.

As result, such libraries are required to assist researchers in completing their jobs more efficiently. Many researchers have been able to obtain data for the purpose of processing and identifying malware, therefore here are some of the studies that have made use of some of the available datasets, as well as their training and testing details. The authors of this paper used the Ember dataset, which is freely available. This collection contains files that might be dangerous as well as helpful. This folder contains 69,860 harmful files and 70,140 benign files. After splitting the dataset into training and testing files, these files are utilised to train the model.

Using Scikit-learn, they partitioned the dataset into 60% for training and 40% for testing. In contrast to the testing dataset, which contained 28.000 acceptable files and 28.000 harmful files training dataset contained 42140 benign files and 41860 harmful files [10]. The authors of another paper attempted to identify the type of malware or the family to which it belonged. In order to analyse the results, they employed the dataset in parts different formats: one set had both benign and dangerous files, and the other set contained just malicious files [12]. The information is presented in Tab. 1.

Dataset	**Malware count**	**Benign count**	**Total**
	1258	37627	38885
MalDozer	20089	37627	57716
All	33066	37627	70693

Table I: Sets of data for identifying malware

The writers attempted to determine whether or not a certain program is harmful. Also, check for the most significant attribute that can assist in defining the malware's family. To support these conclusions, the scientists also employed two additional datasets from Contagio Minidump and Virus Share [13]. This study's authors developed a methodology to identify if a downloaded application is safe or harmful [14]. They used VirusTotal [17], an internet virus detection programme, during the testing process. They used 20,000 malware samples from VirusShare [18], 1,258 malicious files and 37627 innocuous files from the Malgenome project [19], 20,089 malicious files from MalDozer [20], and 37627 benign files from the Google Play App Store to evaluate their work. Tab 2 describes the specifics. The vast majority of Android applications are undoubtedly secure. Because of this fact, it is challenging to detect Android malware, especially when utilizing deep learning techniques.

Dataset	**Malware Count**	Family count
Malgenome	985	9
MalDozer	20089	32

Table II: Datasets for attributes of task

A significant amount of data is needed to identify Android malware, and these data sets must be regularly updating with risky samples. This could help identify Android viruses using deep learning frameworks. Deep learning experts, however, have challenges because the majority of datasets are small.

IV. CONTEXT: TENSORFLOW

TensorFlow is the name of Google's large-scale machine learning model development and deployment technology. Although deep learning is TensorFlow's primary use, it also supports a wide range of other models, such as alternate learning methods. TensorFlow Paradigm: A dataflow graph which describes a computation is a tensorflow model. Nodes in the graph represent various operations. This list includes operations for initialising tensor values as well as constant, sequence, and random

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 08, August-2022

operations, addition and matrix multiplication, summary operations for generating record events for clarity purposes and use variables to maintain model parameters.

TensorFlow Model Building: The TensorFlow API offers high-level methods that the dataflow graph can use to carry out low-level actions. Some people create a lot of low level procedures, like tf.train. Optimum Gradient Descent. Thus, a small amount of code, like the specification of the tf.mnist Simple Model, can produce somewhere between 100 actions in the graph. Networks that are even more intricate exist in the real world.

For instance, a well-known Inception network implementation has more than 36,000 nodes. Three different purposes are served by TensorFlow graph edges. The input and output data for the operations are tensors, or multidimensional arrays, and data dependence edges instantly reflect these tensors. The data flow of the Detection of Smart Android Malware (DSAM), which is based on deep learning and machine learning, is shown in Fig. 4.1(A). For the first stage, the dataset of malicious Android apps will be used. Since the dataset gathers a lot of data about risky, it must first be analysed. The identification of dangerous Android application malware is accomplished using a variety of machine learning approaches in this manner programmes. The display of the malware dataset is aided by data analysis. In order to employ machine learning, data must then be trained and tested.

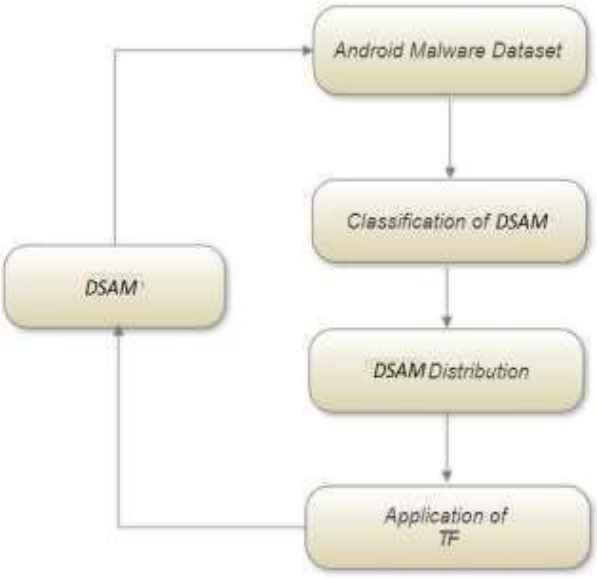


Fig. 4.1 (A) Data Flow Diagram of *DSAM*

Data testing and training produce results that are correct. After that, the tensor flow algorithm will be used. It is an open- source framework for numerical computation that is compatible with Python and speeds up and makes it simple to create neural networks.

Following the use of this method, the percentage-based result will displayed that will detail the app's maliciousness. The outcome will be presented as a percentage and will indicate the quantity of malware foundin the application. which will enable us to determine whether or not the app is safe to use. If the amount of malware is too large, it will advise against using the application and prompt an instant uninstall.

Figure 4.1 (B) explains the Structural Representation of TensorFlow. Using a suite of tools called TensorFlow Lite, machine learning may be done on mobile devices enabling developers to run complex models on embedded, edge, and mobile hardware. Here tf.data is used to read the data for the purpose of processing. Data Importing. From some data, create a Dataset instance. Establish an iterator. Making an Iterator object using the newly created dataset will allow you to run across the dataset.

Data Consumption can acquire the dataset's elements and feed them into the model by using the iterator we constructed And tf.keras enables the design, fit, evaluation, and usage of deep learning models for prediction. A full open-sourced machine learning system is known as TensorFlow. Researchers can advance the state-of-the-art in ML because of its vast, adaptable eco - system of techniques, libraries, and local resources, while developers can simply create and deploy ML- powered apps. It gives common developers who just want to get stuff done access to popular deep learning tasks like classification and regression predictive modelling.

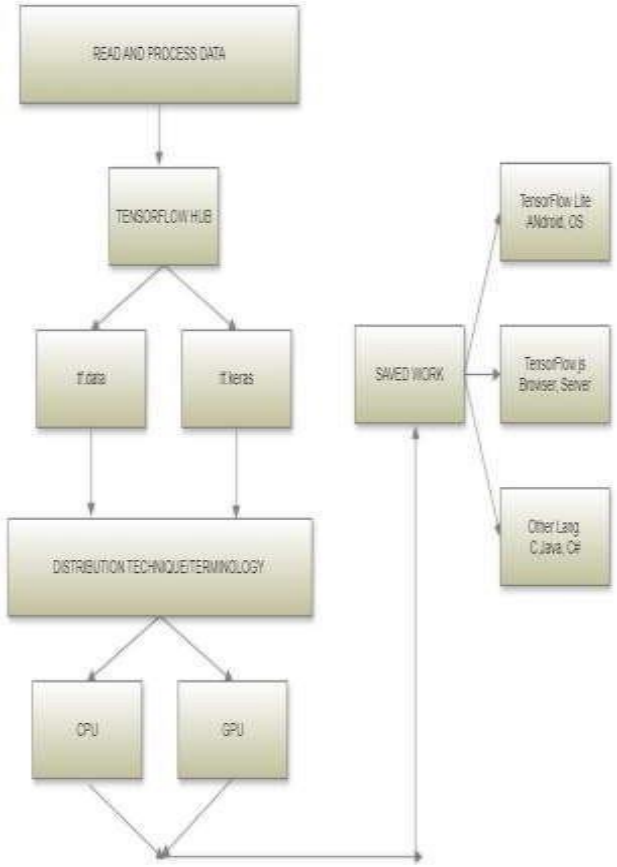


Fig 4.1 (B). Structural Representation of *TF*.

TF.KERAS- Used the high-level deep learning API Keras to put the suggested framework into practise. A neural network library created in Python is called Keras. It is an open source library that utilises Theano and TensorFlow as its foundation. It was created with the intention of enabling users to write their own scripts without having to

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 08, August-2022

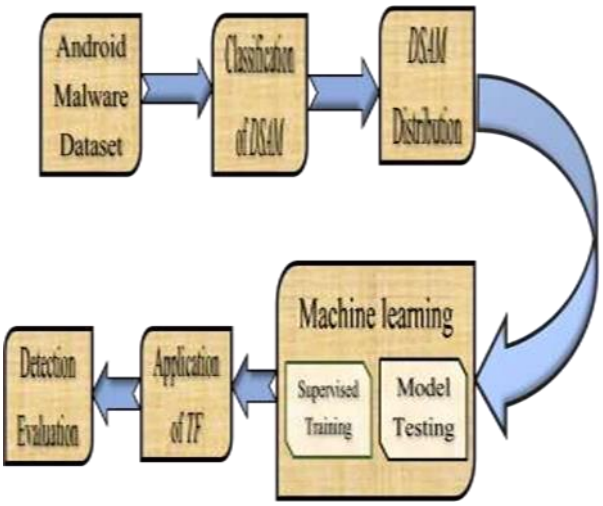
thoroughly understand the backend.
This technology is what we utilised because it makes it simple to experiment quickly with deep neural networks. We selected the keras framework primarily because to the functionality it offers. Some of them are its user-friendly, easily extensible, and modular design. It also offers a simple interface that makes quick and easy prototyping possible. Additionally, it supports every neural network model.

V. PROPOSED METHODOLOGY FOR DETECTION OF SMART ANDROID MALWARE (DSAM) EMPLOYING DEEP LEARNING AND MACHINE LEARNING

Malware is a huge threat to computer users' security, and it may cost businesses a huge amount of money. Detecting malware has become a dynamic challenge for researchers, since their behaviour changes. The two types of malware analysis as well as detection methods are static analysis but also dynamic analysis. The process of running malware and observing its behaviour on the system is known as dynamic analysis. Static analysis is the technique of examining and extracting data from an executable file without actually running it. When a new version of malware is detected, the experts either manually examine the sample or design a programme to match the similarities of malware so it gets detected easily.
To address the above - mentioned security concerns, I propose a "Detection of Smart Android Malware (DSAM) Employing Deep Learning and Machine Learning" paradigm. The identification of dangerous Android application malware is accomplished using a variety of machine learning approaches in this manner. The dataset of malicious Android apps will be used for the initial phase. An analysis of a dataset is necessary first since it compiles a lot of information about dangerous programmes. The display of the malware dataset is aided by data analysis.
In order to employ machine learning, data must then be trained and tested. Data testing and training produce results that are correct. After that, the tensor flow algorithm will be used. It is a Python-friendly open-source framework for numerical computing that accelerates and simplifies the creation of neural networks.
After applying this algorithm, we will be getting the percentage- based result which will elaborate us how the app is malicious. The result will be in percentage which shows us no. of malware present in application. Which will help us to understand whether the app is secure to use or not. If the percentage of malware is too high, then it will suggest not to use the application or immediately uninstall. We offer an evaluation sub-module for comparing based on architectures of deep learning on dynamic analysis with conventional machine learning algorithms (MLAs) for the detection of windows malware. Utilizing behavioural data acquired through dynamic analysis, all models are assessed.
Classification of DSAM: To obtain accurate and thorough information, the dataset is categorised. Data analysis and classification disclose specific information about it, such as the type of data, its size, the number of the same data type, etc.
DSAM Distribution: The dataset is broken down into two groups: 1) benign files and 2) harmful or malware files. The distribution process separates these files. Once the distribution is complete, both sorts of files are subsequently utilised in various ways for malware identification.
Supervised Testing for DSAM: In order to produce accurate predictions, a supervised Learning develops a model to use a predetermined set of input data (the training set) and recognized responses to the data. Regarding how new incoming data will react. If you have previous data for such outcome that are attempting to predict, use supervised learning.

A revolutionary semi-supervised learning method can now be used to find unknown malware.

1. Figure Captions



(Fig. I: shows the architectural diagram of project which explains the malware dataset detection method in stepwise execution.)

The authors have employed the Learning with Local and Global Consistency (LLGC) method, a semi-supervised classification system. In their recommended method, a classifier is learned using a set of labelled (malware and legitimate software) and unlabelled data. Their method could get findings with an accuracy of 0.86 using data that was 50% labelled [4].
Model Testing: We have utilised machine learning techniques, and you have probably come across train test split () or functions with similar names in libraries like scikit-learn, TensorFlow, or others. The first stage in producing accurate predictions is training a model; however, determining the predictive potential is a separate matter. To create a strong foundation on which to train, compare, and test your models, data must be split.
Application of TF: TensorFlow is a scalable and adaptable software library for dataflow graph-based numerical computations. Users can quickly develop, test, and deploy neural network and other machine learning models using this library and related tools. The fundamental algorithms of TensorFlow are created using CUDA and highly

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 08, August-2022

optimised C (Compute Unified Device Architecture). The fundamental components of a TensorFlow programme and demonstrate them using the classification of handwritten digits using a CNN example. High-level APIs, distributed training, and more TensorFlow tools are then introduced [11].

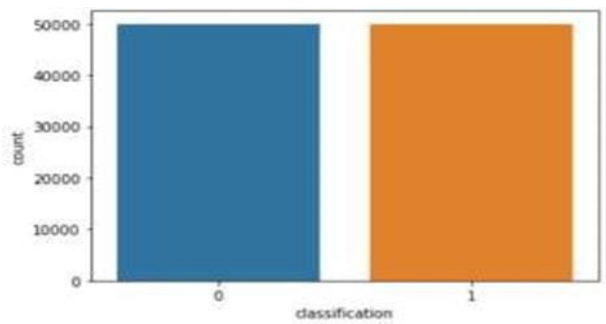
V. EXPERIMENT AND DISCUSSION

Real issues are encountered in the field of cybersecurity, which has direct implications for the general public and some governmental entities. Malware is a hazard to mobile users since it can be downloaded and shared with sensitive and important data. Therefore, malware attacks directed at mobile devices are being carried out by the attackers. The newest methods to detect this infection are currently being developed by researchers. This study provides comprehensive details on the most recent methods and their results in this area.

A well-liked and adaptable machine learning library is TensorFlow. In it are both supports multiple languages interfaces than Python and both low-level and high-level APIs. Users can easily distribute training using the tf.distribute and visualize model and training metrics with TensorBoard. Utilize a strategy and deep neural network technology to incorporate probabilistic methodologies TFP database.

We offered a thorough overview of machine learning-based as well as deep learning methods in roughly the same order as the machine learning design department, for identifying Android malware, and briefly discussed the backdrop to Android malware. At each level, a variety of alternative strategies were considered along with a thorough analysis of their benefits in particular situations. To aid with comprehension and comparative analysis, some essential information was condensed into diagrams and tables.

By focusing more components of machine learning techniques, this work differs from earlier studies on Android virus detection. By addressing some knowledge gaps and bringing up some unresolved concerns in this area, we think this work enhances earlier reviews. We anticipate that this review will serve as a starting point for curious readers and encourage them to explore more research options. Our programme will correctly recognise, detect, classify, and shield Android mobile devices from hazardous apps using an intuitive user interface, preventing any misuse or theft of data.



(Fig 5.1: Classified dataset into two parts.
1) 0 representsBenign files & 2) 1 represents Malware)

This Collection Includes Documents That May Be Both Hazardous and Beneficial. This Folder Has 70,140 Good Files And 69,860 Bad Files. After the dataset has been divided into training and testing files, the model is trained using these files. The training dataset includes 42,140 benign files and 41,860 damaging files compared to the testing dataset has 28,000 entries of harmless records and 28,000 malware ones.

VI. CONCLUSION AND FUTURE WORK

This study discusses numerous machine learning approaches that were used to detect malware threats. However, some of the strategies mentioned here did not disclose the results, some did not cover the characteristics, and others did not display real data set utilized in the investigation.

The use of smartphones is expanding significantly. Smartphones include all of the private and sensitive information. The popularity of Android has significantly increased as well. Future analysis of this data could result in a comprehensive report about the datasets that are currently available, their acquisition details, and thorough systematically organised data regarding feature extraction methods, which would then be categorised based on how closely related the feature information and extraction details are. Then, machine learning methods might be developed further researched to give readers the most recent information.

REFERENCES

[1] Mohammed A. Alqahtani, “Machine Learning Techniques for Malware Detection with Challenges and Future Directions”, in International Journal of Communication Networks and Information Security (IJCNIS) Vol. 13, No. 2, August 2021.

[2] Manish Goyal, Raman Kumar, “A Survey on Malware Classification Using Machine Learning and Deep Learning”, in International Journal of Computer Networks and Applications, 2021.

[3] N. Peiravian and X. Zhu, “Machine Learning for Android Malware Detection Using Permission and API Calls”, IEEE, 2021.

[4] Fakhroddin Noorbehbahani, Mohammad Saberi, “Ransomware Detection with Semi-Supervised Learning”, IEEE,2020

[5] Cline Walter Colaco, Mandar Deepak Bagwe Siddharth Amit Bose, “Detection of Malicious Android Apps Using Machine Learning Techniques”, in International Journal of Trendy Research in Engineering and Technology Volume 4 Issue 7 December 2020.

[6] Kaijun Liu, Shengwei Xu, Guoai Xu, Miao Zhang, Dawei Sun, And Haifeng Liu, “A Review of Android Malware Detection Approaches based on Machine Learning”, in IEEE, 2020.

[7] Garminla Sampath Kumar, Pooja Bagane, “Detection of Malware Using Deep Learning Techniques”, in International Journal of Scientific & Technology Research Volume 9, Issue 01, January 2020.

[8] Zhiqiang Wang, Qian Liu, And Yaping Chi,” Review of Android Malware Detection Based on Deep Learning”, in IEEE 2020.

[9] Muhammad Ijaz, Muhammad Hanif Durad, Maliha Ismail, “Static and Dynamic Malware Analysis Using Machine Learning”, in IEEE 2019.

[10] Vinayakumar R, Mamoun Alazab, Soman Kp, Prabaharan Poornachandran, And Sitalakshmi Venkatraman,“Robust Intelligent Malware Detection Using Deep Learning”, in IEEE 2019.

[11] Bo Pang, Erik Nijkamp, Ying Nian Wu, “Deep Learning with TensorFlow: A Review”, in Journal of Educational and Behavioral Statistics, 2019.

[12] E. B. Karbab, M. Debbabi, A. Derhab and D. Mouheb, “MalDozer: Automatic framework for android malware detection using deep learning,” Digital Investigation, vol. 24, 2018.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 08, August-2022

[13] T. Kim, B. Kang, M. Rho, S. Sezer and E. G. Im, "A multimodal deep learning method for android malware detection using various features," IEEE Transactions on Information Forensics and Security, vol. 14, 2018.
[14] S Ni, Q Qian, R Zhang - Computers & Security, "Malware identification using visualization images and deep learning", ScienceDirect, 2018.
[15] Ayesha Butalia, Swapnil Patil, Manjiri Bangali, Bhushan Dusane, Tejswini Chaure & Maya Ingale, "Computer Science & Engineering: An International Journal", An International Journal CSEIJ, 2017.
[16] Long Wen and Haiyang Yu, "An Android malware detection system based on machine learning", AIP Conference Proceedings, 2017.
[17] "VirusTotal. Accessed: Sep. 2017. [Online]. Available: https://www. virustotal.com/ko."
[18] "VirusShare. Accessed: Sep. 2017. [Online]. Available: https://virusshare. com."
[19] Kanit Wongsuphasawat, Daniel Smilkov, James Wexler, Jimbo Wilson, "Visualizing Dataflow Graphs of Deep Learning Models in TensorFlow", IEEE, 2017.
[20] Fatih Ertam, Galip Aydın, "Data classification with deep learning using Tensorflow", IEEE, 2017.
[21] Xin Su, Dafang Zhang, Wenjia Li, Kai Zhao, "A Deep Learning Approach to Android Malware Feature Learning and Detection", in IEEE, 2016.
[22] A Butalia, AK Ramani, Maya Ingle, Parag Kulkarni, "Classification of Sentiments through Rough Fuzzy Approach", International Journal of Computer Science, Information Technology, & Security.2012
[23] Ayesha Butalia, Manikrao Dhore, Geetika Tewani," Applications of Rough Sets in the Field of Data Mining", in IEEE, 2008.For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].