An Evaluation and Detection of Malicious Mobile Applications with Permission based Approach and Deep Learning

Anurag Rajput
rajpu001@rangers.uwp.edu
University of Wisconsin Parkside
Kenosha, Wisconsin, USA

Dhanashree Ashok Mehta
mehta007@rangers.uwp.edu
University of Wisconsin Parkside
Kenosha, Wisconsin, USA

Daniel Hetzel
hetze005rangers.uwp.edu
University of Wisconsin Parkside
Kenosha, Wisconsin, USA

Adeoluwa D Akinniyi
akinn001@rangers.uwp.edu
University of Wisconsin Parkside
Kenosha, Wisconsin, USA

Susan Lincke
lincke@uwp.edu
University of Wisconsin Parkside
Kenosha, Wisconsin, USA

ABSTRACT

The amount of malware infections on the Android platform has risen in recent years. Our study recorded and compared a statistical analysis between top Google Play Store Android Packages (APKs) and a collection of Malware APKs, which are both available for download (Application Program Interface). We can identify patterns and irregularities in the permissions system based on the information we collect. These applications are decompiled and reloaded the decompiled java into an Android Developer Studio to conduct additional research. We have explored various studies to evaluate permissions from the play store and some have suggested a good evaluation of the permissions actually being run. In our research, we have done a comparative analysis between traditional machine learning algorithms such as Naive Bayes, Decision tree, and Support vector machine (SVM), etc. with deep learning sequential dense layer neural network to distinguish between malicious and benign Android applications (Apps). we have discussed results to show contrast to why deep learning methods are marginally better from a performance perspective than traditional machine learning with an experimental setup and systematic literature review. Finally, we have shown the actual results of our study to guide others in understanding which malware identification technique is efficient.

KEYWORDS

Malware, Android security, Machine Learning, permissions, deep learning

ACM Reference Format:
Anurag Rajput, Dhanashree Ashok Mehta, Daniel Hetzel, Adeoluwa D Akinniyi, and Susan Lincke. 2022. An Evaluation and Detection of Malicious Mobile Applications with Permission based Approach and Deep Learning. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

By 2013, there were more than 500 million mobile subscribers in China[1]. As a result of its widespread use, one of the most popular operating systems, Android, has become a favorite target for malware. In 2011, there was an increase from 80 bad applications to over 400 dangerous applications[2]. Because Android is becoming increasingly targeted and significant, any perceived knowledge of patterns or insights into the permissions that are used to defend the framework are valuable assets to have. To do so, it is necessary to first understand the permissions that are being used, which is the subject of our paper. Permissions that are provided in app shops have been analyzed in multiple papers.Researchers discussed the use of permissions and the permission overload that Android users encounter in the App Store,Consistently high levels of authorization requests, couched in legalese, imply very serious security concerns.They suggests...an overabundance of permissions for Android applications that are actually unnecessary for the features and resource use, as this creates a path for security and privacy concerns." They make theoretical suggestions on which permissions may be more prone to abuse, but they do not have statistics on permission abuse[3]. Some researchers recognizes the possibility of classifying malware based on the permissions it uses, but does not see a direct comparison for protection purposes. Certain permissions contribute to the malware's very evident goal, but neglects to look for patterns that could result in an enhancement in protection[4]. Few of them, utilizes the privilege to scan and evaluate apps with the goal of correctly identifying malware through machine learning[5]. Our objective is to keep track of all instances of certain permissions in an APK, and to contrast and quantify techniques for android malware identification using machine learning and deep learning.

1.1 Deep Learning

Previous studies focused on each model separately and with distinct data sets. Using the identical data sets, we compared six algorithms. They also haven't addressed the deep learning model architecture that is required to solve this challenge; our approach includes this architecture.

1.1.1 DEEP LEARNING DETECTION APPROACHES.

- **Sequential Neural Network:** Many Android malware detection ideas have included sequential neural networks,which is a artificial neural network with dense layer.
- **Convolutional Neural Network (CNN):** Convolutional neural networks group neurons according to their height, width, and depth. The CNN, like other neural networks, has an input layer, a hidden layer, and an output layer. It's a robust deep learning architecture that can deal with high-quality data[6].

- **Recurrent Neural Network (RNN):**There is no relationship between the input and output values in typical neural networks. To put it another way, the output values have no direct impact on the inputs. However, for some applications, the output information is also necessary for the next estimation[6].
- **Long-Short Term Memory (LSTM):** RNNs with long short term memory are a unique sort of RNN. The Long Short Term Memory Network (LSTMN) is a type of sophisticated RNN (sequential network) that permits information to be retained. It can deal with the vanishing gradient problem that RNN has. For persistent memory, a recurrent neural network, also known as RNN, is used[6].
- **Long-Short Term Memory (LSTM):** RNNs with long short term memory are a unique sort of RNN. The Long Short Term Memory Network (LSTMN) is a type of sophisticated RNN (sequential network) that permits information to be retained. It can deal with the vanishing gradient problem that RNN has. For persistent memory, a recurrent neural network, also known as RNN, is used[6].

1.2 Research Question:

The important contributions of this research paper are as follows:

- Systematic literature review to evaluate an appropriate approach for finding malicious application patterns.
- Contrast and analysis of algorithm results, detection of malware using appropriate detection technique.

2 RELATED WORK

For a better understanding of android malware detection analysis, and to know which existing approach is efficient, we have explored multiple research survey papers that have discussed various techniques in their paper.

2.1 ANDROID MALWARE DETECTION APPROACHES

- **Static Analysis:** Static analysis is the process of determining whether or not software is malicious before it is executed.The detection platforms used in static analysis involve string signatures, byte-sequence n-grams, syntactic library cells, control flow graphs, and operational code (opcode) frequency distributions, among others. as a major disadvantage o code obfuscation and dynamic code loading . The executable file must be decrypted and unpacked before performing static analysis. [7]
- **Dynamic Analysis:** Its purpose is to examine harmful code and behavior of the application. When it interacts with the system, this is feasible, and it should be done in a controlled setting. This study has the significant benefit of recording app activities and detecting dynamic code uploading during the run-time operation. Static analysis is difficult to implement due to the overhead of running the application. This paper looks at how to identify Android malware utilizing manifest file permissions, which can aid during the installation in dynamic malware detection.[7]
- **Hybrid Analysis:**: Hybrid analysis of an application combines both static and dynamic analysis. The static analysis is the first step in the procedure. The static analysis examines the application's code, permissions, and components. The dynamic analysis then examines the application's overall behavior. Only a few frameworks, such as Mobile Sand Box and Andrubis, use the Hybrid Approach. [8]

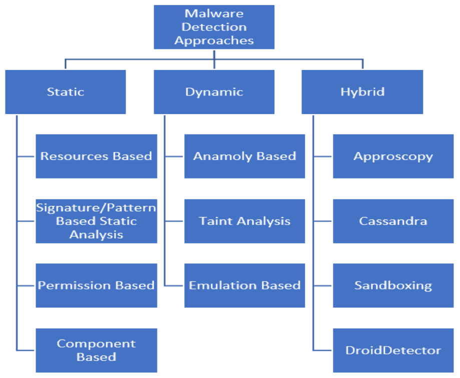


Figure 1: Malware Detection Techniques

P. Agarwal et. al (2019) in their paper described malware detection techniques, they have mentioned static, dynamic, and hybrid analysis approaches but discussed the static analysis approach in detail by defining several static approaches such as Signature/Pattern Based Static Analysis, Resources Based Static Analysis, Components Based Static Analysis, Permission-Based Static Analysis. In their study, they concluded that static analysis is more time, power, and resource-consuming. Also, they lack detection of the run-time behavior of the applications. Also, they cannot detect unknown malware types, and the detection techniques that use dynamic analysis are more resource-consuming and also cannot detect unknown malware types. A hybrid malware detection must be proposed with a machine learning approach that will address restrictions of both static and dynamic analysis. [9]

In P. Agarwal et al study, dynamic and hybrid analysis approaches explanation were missing, hence we have reviewed another paper, which is Zachariah et al. (2017) and In this study they have revealed numerous dynamic and hybrid analysis approaches such as Anomaly-based detection, Taint Analysis, and Emulation based Detection for hybrid analysis techniques such as Approscopy, Cassandra, DroidDetector, and Sandboxing. They have also mentioned results but without discussing their experimental setup. Finally, they have concluded that static approaches are generally faster, provide higher accuracy rates, and are effective against existing malware attacks whereas dynamic approaches perform well and can detect malware variants and uses a real-time analysis, and the hybrid method provides much better performance but are expensive because of the limited resources, Hence as per their study cessation, no single approach is enough to make a system secure. [10]

Our project is doing static analysis with permission based approach.

2.2 MACHINE LEARNING DETECTION APPROACHES

Machine learning techniques were used to categorize Android apps proposed by the author in [4]. From the Android market, the permissions of the app are also gathered, as are the printable strings' frequency, the app's permissions and the various permissions of the app itself as they appear in the Android market. Researchers of the paper [4] tested 820 samples from seven different families and found the Bayes TAN to be the best classifier by using J48, Random Forest, KNN, Na've Bayes, SVM and Bayesian Networks as classifiers.[11] [12]

There were two machine learning assisted approaches presented in this thesis [3] for static analysis of Android malware. The first methods is based on permission features. A logistic regression model was used as a classifier and delivered Precision of 0.823, Recall of 0.822 and F-Scores of 0.821. An alternative approach is to extract features from code files. Android apps are rewritten into multiple Java files, and then feature vectors are generated using natural language processing using the bag-of-words model. The framework is capable of integrating SVM with SMO, simple logistic regression, AdaBoostM1 and logistic regression with SVM, and each has achieved F-scores of 0.956, Precision and recall of 0.958 and 0.957, respectively.[13][14]

A Bayesian classification method has been proposed to detect the Android malware. Three tools were developed by the author [2] : command detectors, API call detectors and permissions detectors. In their experiments, top-n attributes are selected that are discriminative in nature that form effective features based on resources, API calls, libraries, assets, and permissions, respectively. A Bayesian classifier is finally trained, which consists of 1000 samples of malware and 1000 samples of benign applications. Using 20 of the classification attributes, performance reaches the Accuracy of 0.921, Precision of 0.935, and AUC 0.97223.[15]

3 METHODOLOGY

To gain a deeper understanding of effective malware detection techniques, we have done a comparative analysis between traditional machine learning techniques(Naive Bayes, Decision tree, Random Forest, Support Vector Machine, K-Nearest Neighbours and deep learning artificial neural network.

Process flowchart depicted in figure 2, showing methodology of comparative analysis.

3.1 Data Collection

In our study, we employed two data sets, dataset-A and dataset-B, which we gathered from kaggle and other internet resources.We use dataset-A to train and validate of our model, and then dataset-B to contrast our machine learning and deep learning techniques.

- Dataset-A have 175 features collected from kaggle that reflect mobile permission or tasks used by applications and 15036 observations reflecting applications. All of the variables are categorical in nature and we used them all comparison analysis.
- Dataset-B is have 175 features collected from multiple online resources and 221 observations reflecting permission and applications respectively.

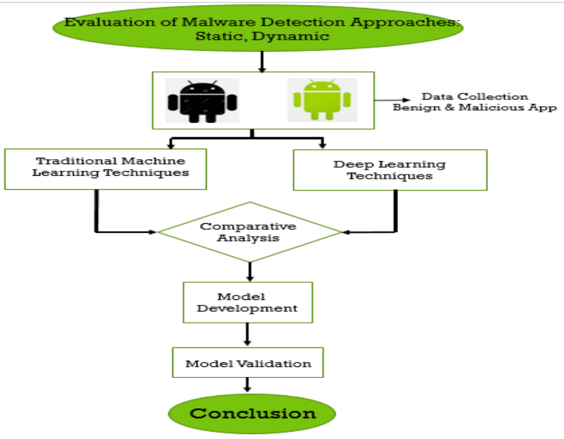

Evaluation of Malware Detection Approaches: Static, Dynamic
Data Collection Benign & Malicious App
Traditional Machine Learning Techniques
Deep Learning Techniques
Comparative Analysis
Model Development
Model Validation
Conclusion

Figure 2: Comparative Analysis Process Flow

3.2 Variable Selection

- Dependent Variable: Our dependent variable has two labels in our study, making it a binary classification: if an application is malware, it is labeled as 1, and if it is benign, it is categorized as 0.
- Independent Variables Selection: We needed an independent variable collection that could provide as much information as permissions-based information for Android malware detection, thus we used all variables from the dataset-A.

3.3 Evaluation Techniques

The data sets are compared and evaluated using five machine learning techniques: Navies Bayes, Support Vector Machine, Random Forest, K-nearest neighbor,Decision Tree Algorithm and Deep Learning. To compute the accuracy of presented experiments we have computed the confusion matrix in order to evaluate the efficiency of the data. The confusion matrix uses the following metrics:

- **True Positive (TP):**Number of non-malware i.e benign applications that have been correctly identified.
- **False positive (FP):** The number of malware applications which are identified correctly.
- **True negative (TN):** Total number of malware applications which are correctly identified.
- **False negative (FN):** Number of non-malware i.e benign applications which are wrongly identified.
- **True positive rate (TPR):**It is a Percentage of correct identification of goodware i.e benign applications. (TP/(TP+FN))
- **False positive rate (FPR):**It is a Percentage of wrong identification of malware applications. (FP/(TN+FP))
- **Accuracy:** Correctly identifying the percentage of applications.

$$\textbf{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

All the five classifiers are presented experimentally in Table 1 for a number of data sets. It can be seen that the Naive Bayes classifier performed better than other algorithms. It was noted that the accuracy averaged 99.96 percent, the highest of all classifiers.

Table 1: Tools and Techniques used by studies

S.No	Name	Keywords	Tools/Techniques Used
1	S.Sabhadiya et. al.	Android Security, Malware Detection Technique, Deep Learning based Malware Detection	Maldozer, DroidDetector, DroidDeepLearner, DeepFlow, Droid Delver and Droid Deep
2	Xu, Ke and Li et. al.	ICC, malware detection, Android.	Inter-component communication (ICC) mechanism

3	Suleiman Y and Igor Muttik	Mobile security;
		Android; malware detection;
		bayesian classification;
		static analysis; machine learning;
		data mining;
			Bayesian Classification Model

4	Sanz Borja et. al.	Humanoid robots, Androids, Feature Extraction, Malware, Machine Learning, Learning Systems,Training	Machine Learning Classifiers
5	Burak and Beyzanur	Malware Detection; Mobile Malwares; Static Code Analysis ; Machine Learning; Artificial Neural Neworks; Random Forest	Random Forest, Deep Learning
6	H. Soni et. al.	Significant Permission Identification, Decision Tree Algorithm, Random Forest Algorithm, Machine Learning	Machine Leraning algorithms like Random Forest and Decision Tree
7	Neha Tarar et. al.	android; android malware; static analysis; opcodes; feature selection; machine learning; classification	Machin Learning Classification
8	Prerna and Bhushan	Smartphones, Android, Malware, Detection Techniques, Mobile Security.	Static, Dynamic and Hybrid
9	Xin Su et. al.	Android malware, Deep learning, Security, Static analysis	DroidDeep
10	Hadiprakoso et. al.	malware analysis, hybrid-based analysis, android malware, malware detection, deep learning	AutoDroid,DNN,Deep4MalDroid
11	Bourebaa and Mohamed	Deep learning, Convolutional neural networks, Mobile security, Android malware detection	Convolution Neural Networks

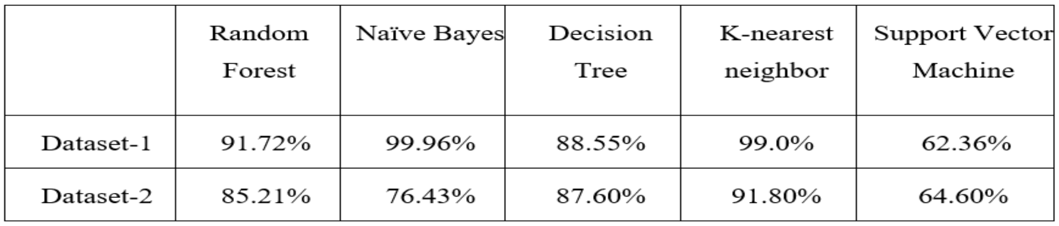
12	Sandeep HR	Malware Classification; Malware; Machine learning;
		Security; Android;
		Permissions; APK (application package); Deep learning;
		Data Mining; Data Extraction;
		Preprocessing; Vector Representation;
		Behavioral Analysis; Keras;
		Deep Learning Dense Model; Random Forest
		Classifier; Virus Share
			Machine learning Classifiers and
			Deep Learning Models

13	TaeGuen et. al.	Android malware, malware detection, intrusion detection, machine learning, neural network.	Multimodal Deep Architecture
14	Jianming and Futai	Android malware, malware detection, intrusion detection, machine learning, neural network.	Recurrent Neural Network
15	Robert and Grzegorz	Mobile systems, android, security, artificial intelligence, machine learning, cybersecurity.	DNN,CNN,Random Forest

In terms of computational complexity, the Na've Bayes classifier performed better. The K-nearest neighbour classifier performed very close to the Naive bayes classifier in android malware detection. Its computational complexity is poor, however, compared with the Support Vector Machine(SVM) classifier.

4 TRADITIONAL MACHINE LEARNING TECHNIQUES

- **Support Vector Machine (SVM) Algorithm:**SVM stands for Support Vector Machine and is a supervised machine learning technology that can address both regression and classification problems. Each data item is represented as a point in n-dimensional space (where n is the number of characteristics we have), with the coordinate value of each



	Random Forest	Naïve Bayes	Decision Tree	K-nearest neighbor	Support Vector Machine
Dataset-1	91.72%	99.96%	88.55%	99.0%	62.36%
Dataset-2	85.21%	76.43%	87.60%	91.80%	64.60%

Figure 3: Performance evaluation of classifiers

feature. Then we utilize classification to distinguish between the two groups. SVM is effective when the number of dimensions exceeds the number of samples. [14]

- **Naïve Bayes Algorithm:**In comparison to numerical input variables, the Naive Bayes classification algorithm works well with categorical input data. It can be used to make predictions and forecast data using past data. Implementation is easy. The conditional probabilities can be easily accessed. Probabilities can be estimated immediately therefore iterations are not necessary. Whenever training speed is essential this strategy is used[15].
- **K-Nearest Neighbors (KNN) Algorithm:**KNN is known as a lazy classification model or lazy learner because when we submit the training data it does not perform any training on that data. Instead of that, it just records the data and does not do any computations during the training period. It does not begin modeling until the dataset is queried. KNN employs a voting mechanism to determine the class of an unseen observation. In our dataset we have used total 4 k values as 3,12, 6 and 6. This means that the class of the data point will be selected by the class that receives the most votes. If K is equal to one, we shall only use the data point's nearest neighbor to estimate its class. We'll use the five nearest neighbors if K equals five, and so on[14].
- **Random Forest Algorithm:**You can partition data, give it to several decision trees, merge multiple trees into a forest, and utilize majority of the votes to find the most appropriate option with random forest. Random forest is a form of group made up of accumulating decisions (outcomes) from several algorithms. To produce a random forecast, a vast number of decision trees are created. Each decision tree predicts a value and then chooses the average of the predicted values. The first is to design a tree, and the second is to get the tree to forecast. Almost every tree in the group is constructed from a sample selected from the training set with replacement[14].
- **Decision Tree Algorithm:**It is an algorithm used in data analytic that uses conditionally control statements which are used to classify data. A decision tree is a diagram that looks like a flowchart a clear path to a conclusion in its most basic form. A decision tree begins with a single node (or 'node') and then branches in two or more ways. The branch presents a number of alternative outcomes, integrating a range of decisions and random events until a final result is reached. Before using, only minor preparation or data cleansing is required[14].

4.1 Result and Analysis:

Figure 1 depicts a graphical representation of all the classifiers that are being compared in terms of classification accuracy. Naïve bayes works best for Kaggle dataset with the accuracy of 99.96 percent but for our dataset the accuracy by Naïve bayes is 76.43 percent. Support Vector Machine(SVM) provides very poor results for both the datasets.

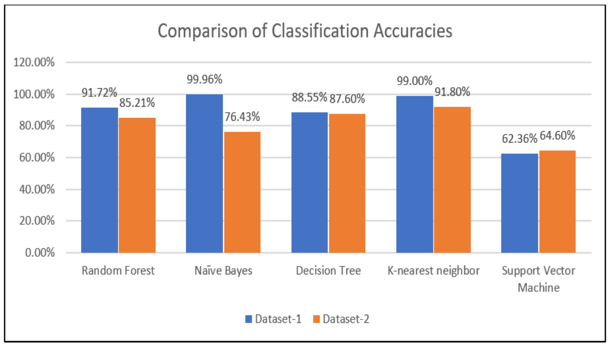


Figure 4: Graphical Representations of Performance evaluation

5 DEEP LEARNING SEQUENTIAL MODEL TECHNIQUE:

We have used Keras dense layer sequential model for our study.A sequential model is a stack of dense layers with exactly one input tensor and one output tensor for each layers, In other words information flows sequentially from the input layer into the hidden layer, into the output layer for sequential modelling.While designing the dense layers of sequential model, we must need to choose carefully architecture of sequential model(number of neurons layers stack), and parameters such as activation function, number of channels according to problem or dataset complexity etc. The terminology used in the deep learning model is defined below.

- **Dense Layer:** Dense layer is a densely connected neural network layer, meaning that each neuron in the dense layer receives input from all neurons in the previous layer. In the models, the dense layer is revealed to be the most usually used layer.
- **Neuron:** A layer is made up of microscopic units known as neurons. Biological neurons can help us understand a neuron in a neural network better. A biological neuron is analogous to an artificial neuron. It takes in information from other neurons, processes it, and then provides an output.

- **Activation Function:** An activation function is a function that is engaged if a set of input features meets the activation function's threshold value, and it is used to help an artificial neural network learn complicated patterns from data. An activation function's ability to add non-linearity to a neural network is its most important feature.

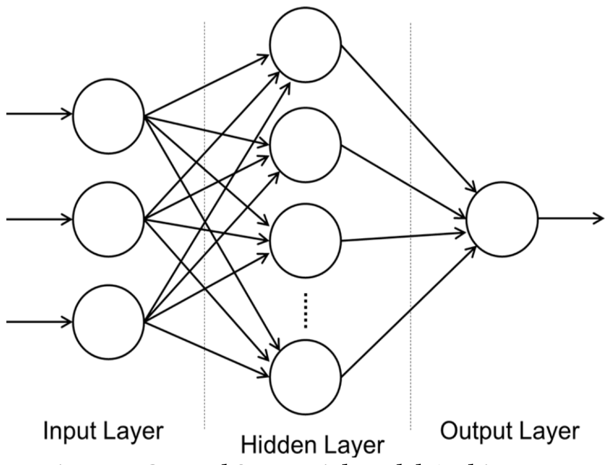


Figure 5: General Sequential Model Architecture

5.1 Model Architecture Setup:

Below is actual sequential layers setup,we have used to make our model.

- **Input Layer:**This is the first layer of any sequential model it contains, number of neurons,activation function, and input Array Size.
  - **Number of Neurons:** 4
  - **Activation Function:** Rectified Linear Activation Function (Relu)
  - **Input Array Size:** (174,)
- **First Hidden Layer:** Hidden layers contains only activation function and neurons.
  - **Number of Neurons:** 4
  - **Activation Function:** Rectified Linear Activation Function (Relu)
- **Second Hidden Layer:**This Hidden layer contains only activation function and neurons.
  - **Number of Neurons:** 4
  - **Activation Function:** Rectified Linear Activation Function (Relu)
- **Output Layer:** Output layer also contains only activation and number of neurons according to type of problem.
  - **Number of Neurons:** 2 (As we have only two classes benign and Malware)
  - **Activation Function:** Softmax

5.2 Model Parameters while Compiling:

- **Optimizer:** Optimizers are algorithms used to minimize an loss function. these are mathematical functions that are based on the learnable parameters of a model, such as Weights and Biases. Optimizers helps in determining how to adjust the weights and learning rate of a neural network in order to minimise losses.
  There are multiple optimizers defined but we have used adam optimizer because it gives faster computation results as compared to others optimizers.
- **Loss Function:** It is simply a measure of fitness of model on given dataset, it calculates the difference between the target's actual and expected value. In our study we have used sparse categorical cross-entropy.
- **Metrics:** Accuracy is number of correct predictions / total number of predictions.It is used to evaluate model classification performance.

5.3 Model Fitting:

It is a important step in deep learning models, at this stage we need to carefully define epochs and batch size for model compilation.

- **Batch Size:** A hyperparameter that specifies how many samples must be processed before the internal model parameters are updated.In our study we have kept batch size 32.
- **Epochs:**The number of epochs is a hyperparameter that specifies how many times the learning algorithm will iterate over the whole training dataset. We have used 10 epochs.

6 RESULTS AND FINDINGS:

6.1 Dataset-A:

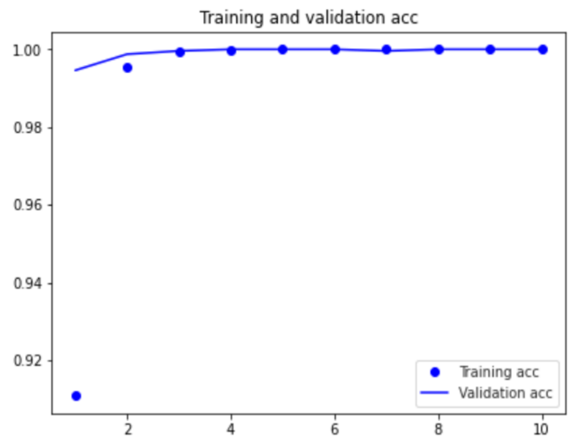


Figure 6: Training Accuracy and Validation Accuracy

6.2 Dataset-B:

6.3 FINDINGS:

As shown in figure 7 and 8, our training and testing accuracy and loss are overlapping that indicates model would give same results(accuracy) in same distributed dataset and in other words our model is pretty much generalized.

To check the performance of our model,we have used dataset-B and figure 9 and 10 shows the results of it. In intial epochs there is difference between acuracy and loss curve because model updated

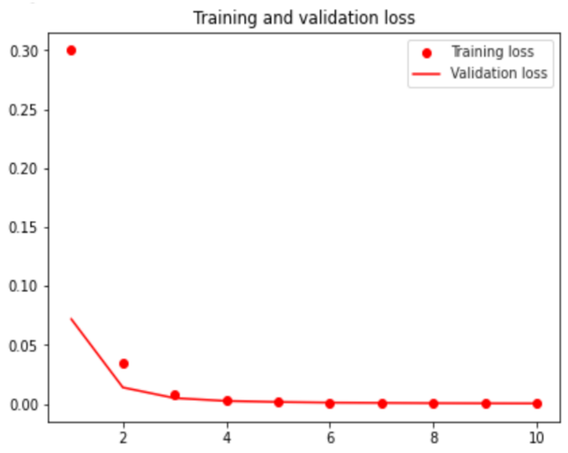


Figure 7: Training Accuracy and Validation Loss

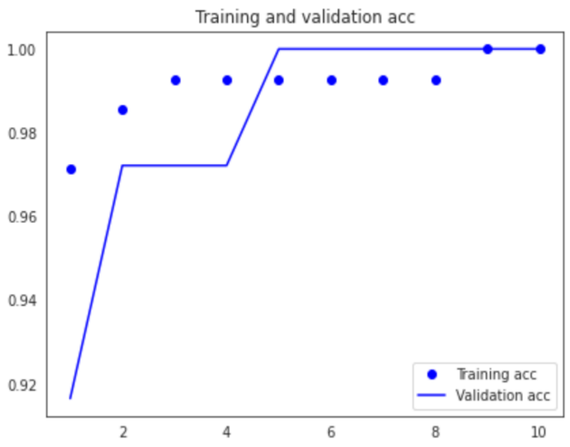


Figure 8: Training Accuracy and Validation Accuracy

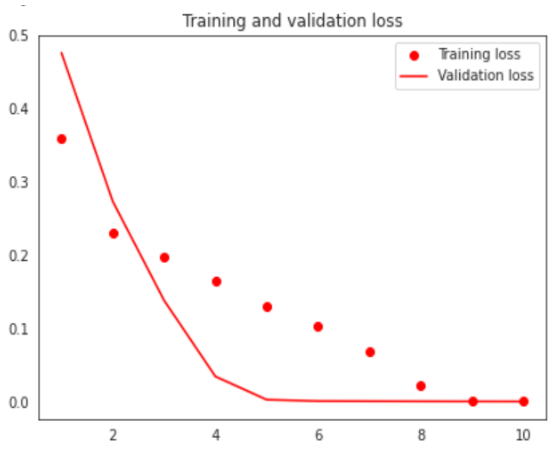


Figure 9: Training Accuracy and Validation Loss

their weights as per new dataset and later on after epoch 9, we can see results similar to dataset-A, that signifies our model is best fit to identify malware applications.

7 CONCLUSION AND FUTURE WORK:

We have contrasted and quantified various Machine learning techniques with deep learning to demonstrate the performance of each same data-sets while deep learning is marginally better from a performance perspective relative to Machine learning however Deep learning is more precise and consistent across data-sets than machine learning. To support our study apart from model development training and testing, we had feed other 200 malware applications data, which was collected from other internet sources, and we found again good accuracy with deep learning model. Hence we concluded that deep learning sequential model is efficient to detect malware applications. For Future work, we will use other malware pattern evaluation approaches to make our model more robust for any type of malwares.Also, we will use statistical measure to distinct which permission is important to identify malwares.

REFERENCES

[1] (2020) Idc: Smartphone shipments will increase by 40 percepts in china next year. [Online]. Available: http://news.xinhuanet.com/tech/2012-12/18/c_124111047.htm

[2] D. Heger, "Mobile devices - an introduction to the android operating environment design, architecture , and performance implications," 2011.

[3] N. Chiluka, A. K. Singh, and R. Eswarawaka, "Privacy and security issues due to permissions glut in android system," in *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, 2018, pp. 406–411.

[4] E. Egemen, E. İnal, and A. Levi, "Mobile malware classification based on permission data," in *2015 23nd Signal Processing and Communications Applications Conference (SIU)*, 2015, pp. 1529–1532.

[5] P. R. K. Varma, K. P. Raj, and K. S. Raju, "Android mobile security by detecting and classification of malware based on permissions using machine learning algorithms," in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2017, pp. 294–299.

[6] M. Uğurlu and A. Doğru, "A survey on deep learning based intrusion detection system," in *2019 4th International Conference on Computer Science and Engineering (UBMK)*, 2019, pp. 223–228.

[7] S. HR, "Static analysis of android malware detection using deep learning," in *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, 2019, pp. 841–845.

[8] P. Agrawal and B. Trivedi, "A survey on android malware and their detection techniques," in *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, 2019, pp. 1–6.

[9] S. Sabhadiya, J. Barad, and J. Gheewala, "Android malware detection using deep learning," in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, 2019, pp. 1254–1260.

[10] R. B. Hadiprakoso, I. K. S. Buana, and Y. R. Pramadi, "Android malware detection using hybrid-based analysis amp; deep neural network," in *2020 3rd International Conference on Information and Communications Technology (ICOIACT)*, 2020, pp. 252–256.

[11] K. Xu, Y. Li, and R. H. Deng, "Iccdetector: Icc-based malware detection on android," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1252–1264, 2016.

[12] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, and P. G. Bringas, "On the automatic categorisation of android applications," in *2012 IEEE Consumer Communications and Networking Conference (CCNC)*, 2012, pp. 149–153.

[13] B. TAHTACI and B. CANBAY, "Android malware detection using machine learning," in *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*, 2020, pp. 1–6.

[14] H. Soni, P. Arora, and D. Rajeswari, "Malicious application detection in android using machine learning," in *2020 International Conference on Communication and Signal Processing (ICCSP)*, 2020, pp. 0846–0848.

[15] S. Y. Yerima, S. Sezer, G. McWilliams, and I. Muttik, "A new android malware detection approach using bayesian classification," in *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, 2013, pp. 121–128.