International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Malware Detection and Classification on Different Dataset by Hybridization of CNN and Machine Learning

S. Arshad Hashmi

Submitted: 25/09/2023 **Revised**: 15/11/2023 **Accepted**: 25/11/2023

Abstract: Malware has long been employed in cyberattacks. Due to their widespread usage, malicious software developers target Android smartphones, which may store a lot of sensitive data. As the main mobile OS, Android has always attracted malware developers. Thus, several Android malware species target susceptible people everyday, making manual malware analysis unfeasible. ML and DL methods for malware identification and categorization might help cyber forensic investigators curb the spread of malicious software. Applying DL methods helps safeguard applications. Cybersecurity issues including intrusion detection, malware classification and identification, phishing and spam detection, and spam recognition have been addressed using DL approaches. ECNN uses the BP (Back Propagation) model for every layer between several intermediate layers, making it faster and more accurate than other methods. SVM Learning with Weighted Features and CNN with SGD optimization for static analysis of mobile apps are presented in this research. The ECNN model has the highest accuracy of 96.92, 96.14, and 95.8 for Android Malware Dataset-1, 2, and 3. On the three datasets, the ECNN model has 96%, 94%, and 94% precision. Smartphone malware analysis is faster and more accurate using this method.

Keywords: *Android, Cybersecurity, Deep learning, Malware, Machine learnings Optimization, Weighted Features*.

1. Introduction

Apache For the next innovative generation, Android OS has become the principal operating system to be used in mobile devices. Subsequently, the activities of Android malware also shot up **[1].** The rapid growth of smart mobile devices and applications implied with Android OS has gained importance due to its feasibility. On the other side, it also faces more security risks. Hostile applications stealing customers' private information, exploiting the prerogative increase to control the device, forwarding text messages to initiate deductions, etc., significantly harm the end user's property. In order to identify Android malware, software developers and researchers propose various approaches based on ML techniques implied with static attributes of the apps, which are considered input vectors that consist of apparent benefits in operational efficiency, code coverage, and massive sample detection **[2**]. Malware is software cybercriminals create to gain unauthorized access to a computer or network or harm the target without their knowledge [3]. Multiple types of malware have witnessed a metamorphosis, resulting in the emergence of hybrid forms with similar attack capabilities. Logic explosives, for instance, are pre-programmed assaults that the victims themselves frequently trigger. In addition, deception and social engineering are employed to deliver malware directly to unsuspecting users. In addition, there is an emphasis on mobile malware, which targets mobile devices in particular [4-7].

Department of Information Systems, Faculty of Computing, and Information Technology in Rabigh (FCITR), King Abdulaziz University, Jeddah, 21911, Saudi Arabia
Email : ahsyed@kau.edu.sa

DL dominates the world of the internet in various computer-related tasks. DL techniques not only allow rapid progress in the rivalry, but it also surpasses human performance. One such task is Image Classification. Disparate in conventional methods of ML approaches, DL classifiers are trained using feature learning alternatives to task-specific strategies. Shortly, it is stated that automatic extraction of attributes and classification of data into different classes [8,9].

1.1. Motivation And Challenges

I. • The Android security system does not restrict the application's usage of the device's resources, including the RAM and the Central Processing Unit (CPU). This is a critical weakness that malicious programs might exploit.

II. • Android's underlying framework for security is very dependent on permission-based processes. When users begin installing an application on an Android device, they are informed of the permissions that the program needs.

III. • Traditional anti-virus solutions cannot protect users against ever-increasing malware dangers since they depend on signature-based detection methods. Static analysis is a method for identifying Android malware that is low-cost and high-performance. Despite these benefits, static analysis has hurdles and limitations, such as dynamic code loading and obfuscation methods.

IV. • The methodologies that were focused on static analysis did not conduct a comprehensive review of the native code. Dynamic analysis techniques have a more significant processing cost and are more sensitive to risk variables than traditional methods.

V.• To comprehensively investigate the application's behaviors, these methodologies need to cover all possible execution pathways. It is possible to disregard some execution routes because most User Interface (UI) triggering mechanisms, such as Monkey Runner, produce random events while dealing with apps. In addition, there is no way to ensure this problem will ultimately be fixed.

• To avoid discovery, attackers often used anti-emulation or analysis-aware tactics. Suppose the malicious software detects that it is being executed in an environment designed for analysis. In that case, it will disguise its malicious behavior and carry out harmless tasks to avoid being discovered.

• Some malicious software programs attempt to conceal their harmful actions for a while. Most methods for detecting malware on Android disregard analytical evasion tactics such as code encryption, repackaging, native code, and dynamic code loading. The accuracy of detection algorithms based on machine learning decreases with time, and feature selection procedures are not robust to changes in the source code.

- The currently available detection methods based on machine learning did not study which characteristics are most valuable for efficiently differentiating Android malware. Despite the development of a wide variety of ways for detecting and analyzing malware, the detection performance of new malware continues to be a significant problem.
- The organization of this research article is as follows: The second section provides related work on malware detection and current research gaps. The methodology and its details are described in the section third. It discusses the proposed MDS methodology in detail, which briefly discusses the data loading process, pre-processing details, feature selection, and CNN implementation details. This section also contains the dataset description. Section four details performance metrics and the results, analysis, and discussions of the experimental observations. Finally, the present study is concluded in the last section 5.

2. Literature Review

Recent Android malware detection attempts use innovative machine learning (ML) and deep learning (DL) methods. A few recent ML/DL studies are listed below. Finally, Table 1 summarizes open-access research on classifying Android malware detection paradigms using several machine-learning methods.

Their research [10] explored an autonomous Android malware detection and attribution system. This system classifies patterns using deep learning methods. Deep learning has successfully identified and classified malware by family. From benign and malicious datasets, Suleiman et al. [11] collected static features including permissions, intents, API calls, and appearance date. To evaluate Naive Bayes (NB), J48, Support Vector Machine (SVM), Random Forest (RF), and Simple Logical classifiers. Garcia et al. [12] presented machine learning-based Android malware family detection. Classified Android API use, reflection, and native application binaries are used to choose system attributes. A hybrid analysis technique with 77 optimum characteristics was reported by Dhanya and Kumar [13]. Teubert et al. [14] suggested a machine learning-based application screening method including dynamic and static analysis.

Table 1. Related Studies on Malware detection

Tool/	Objective	Methodology					
Technique	Detection						
		Anal- Esti-	Static	Dy- Signature System Calls /	Virtual	Machine	Deep
		ysis mation		namic Based API Based	Machine Emulation	Learning Based	Learning Based
DexRay [15]	✓	✓ ✓		✓ ✓			✓
Famd[16]		✓ ✓		✓ ✓	✓		
Ad- Droid [17]	✓	✓	✓	✓			
JOWM- Droid [18]	✓	✓		✓ ✓	✓	✓	

Deep- AMD [19]	✓	✓	✓	✓		✓	✓
DAMBA [20]	✓	✓ ✓	✓	✓ ✓ ✓		✓	
AdMAt [2]	✓	✓	✓	✓	✓	✓	
GDroid [21]	✓	✓ ✓	✓		✓	✓	
Kron- oDroid [22]	✓	✓ ✓		✓ ✓	✓		
Pro- Droid [1]	✓	✓	✓		✓	✓	
Hawk [8]	✓	✓ ✓	✓	✓	✓	✓	
LinReg- Droid [7]	✓	✓	✓	✓	✓	✓	✓
FAM [23]	✓	✓ ✓	✓		✓	✓	✓
MDTA [9]	✓	✓ ✓			✓	✓	
Droid- Fax [24]	✓	✓ ✓		✓ ✓ ✓	✓	✓	
maDroid [25]	✓	✓ ✓		✓ ✓ ✓			
DroidE- volver [26]	✓	✓ ✓	✓	✓ ✓ ✓			
Androct[27]	✓	✓ ✓	✓	✓ ✓ ✓			

2.1. GAP ANALYSIS

• Restricted complexity in terms of the design of the features, the classification algorithm, and the cost of training the extraction of one-of-a-kind structural features, which are more computationally efficient than content-based features, and the extraction of behavioral dynamic characteristics permits the classification of obfuscated, metamorphic, and packed malware without the need for any deobfuscation or unpacking of the malware [16].

• Static and dynamic code analysis, in which several different machine learning methods or PHMM are utilized to investigate the combined static and dynamic aspects of an application in real-time [17,23]

• Its adaptability for industrial applications, which is particularly important given the need to strike a balance between complexity and performance [24,25]

• Determining important common traits shared by malware samples belonging to a certain class to categorize different kinds of malware [28].

• The implementation of more localized feature extraction methods that consider the fundamental segments, in addition to one-of-a-kind image-based characteristics, such as the regional and geographical distribution of texture patterns of malware binaries, resolving data inconsistencies using strategies that are sensitive to costs [29].

3. Methodology

Using an Enhanced Convolutional Neural Network (CNN) and Support Vector Machine (SVM) approach to train our model, this paper describes an improved and effective method for malware detection. Figure 1 depicts the framework for the proposed model, while Figure 2 depicts the proposed Convolutional Neural Network (CNN) architecture. A systematic approach is required to effectively train the malware dataset and then apply it to our classification problem. Figure 3 depicts the comprehensive workflow in its entirety. This study classifies an Android malware dataset utilizing an improved convolutional neural network with SGP optimization techniques in the pooling layer and hyperparameter tuning. Moreover, we have incorporated Weighted Features to improve Support Vector Machine Learning (ESVM). Further, elucidation on the topic as mentioned earlier will be provided in the subsequent section.

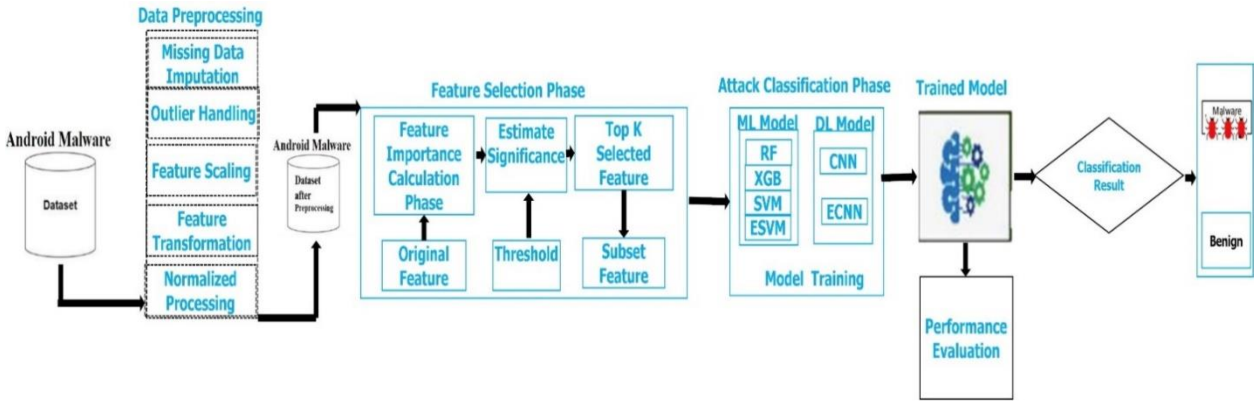

Fig 1. Proposed framework for Android Malware Detection

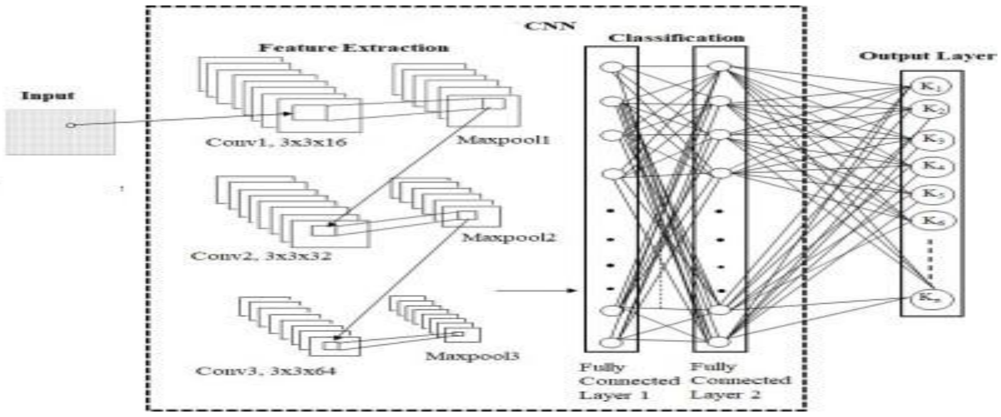

Fig 2. Proposed CNN Architecture

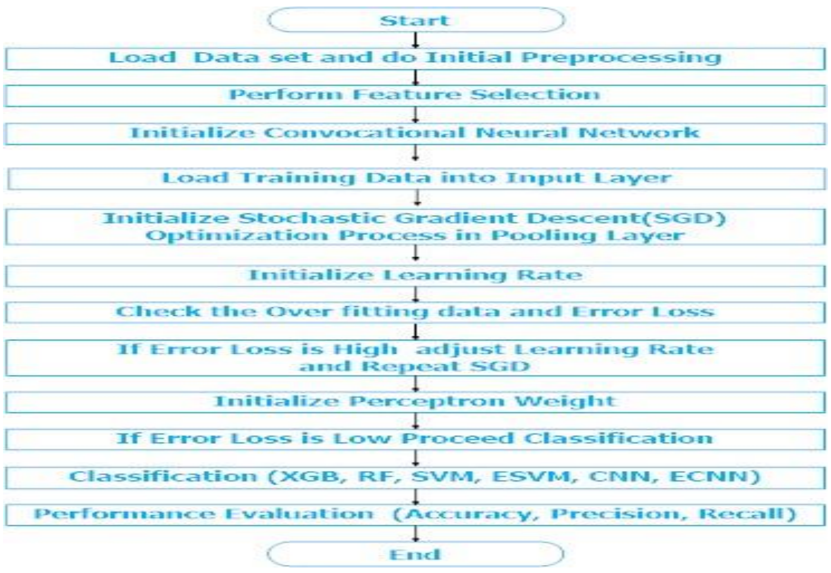
Start
Load Data set and do Initial Preprocessing
Perform Feature Selection
Initialize Convocational Neural Network
Load Training Data into Input Layer
Initialize Stochastic Gradient Descent(SGD) Optimization Process in Pooling Layer
Initialize Learning Rate
Check the Over fitting data and Error Loss
If Error Loss is High adjust Learning Rate and Repeat SGD
Initialize Perceptron Weight
If Error Loss is Low Proceed Classification
Classification (XGB, RF, SVM, ESVM, CNN, ECNN)
Performance Evaluation (Accuracy, Precision, Recall)
End

Fig 3. Work Flow for Android Malware Detection

Android-powered smartphones are very popular because of their excellent performance and open-source nature. Despite this, malware is more likely to be created on the Android platform because of how familiar it is. Traditional signature-based methods for detecting malicious software cannot recognize unfamiliar apps. Many security-focused settings have found DL techniques to be useful for securing apps. In the realm of cybersecurity, DL techniques have been used to pressing problems including intrusion detection, malware categorization, spam detection, and phishing identification. The suggested framework is briefly outlined in the next section.

A. Data Loading

When we talk about "loading data," we're referring to the process of sifting through a database and applying a loading strategy. Data is often sent to the target application in a format that differs from the original source. The ETL process consists of three distinct phases: data retrieval, transformation, and loading into an output container. One or more inputs may be used to generate results at several destinations. In all, three datasets from the public domain were utilized to conduct this research. Information on the dataset itself may be found in subsections 3.3, 4.3, and 4.4.

B. Pre-processing

Raw data is transformed into a form that can be read and analyzed by computers and ML in the data pre-processing stage, which also includes data mining and analysis operations. Before being fed to the model for training, the datasets utilized in this work underwent a number of pre-processing steps, including imputing missing values, outlier treatment, feature scaling, and feature modification.

C. Feature selection method

Android malware detection analysis can be divided into two primary categories: static and dynamic analysis [30]. The static analysis process involves analyzing the source code and associated program resources without executing the program [31]. The dynamic analysis method is utilized to observe and analyze the functioning characteristics of an Android application. Dynamic analysis provides several advantages that cannot be attained through static analysis. First, dynamic analysis can detect malicious activity that static analysis methods cannot. In addition, it is equipped to manage malicious code that employs obfuscation technology effectively. Moreover, dynamic analysis can evaluate an application's efficacy even when critical components, such as a signature, are absent. However, the process of analysis and detection in this context requires a significant allocation of resources because it is more time-consuming than static processes. Utilizing dynamic analysis offers the advantage of an extensive feature selection space and various input classifiers. The third category of Android malware detection is hybrid analysis. Hybrid analysis combines static and dynamic data to distinguish between benign and malevolent programs [32]

D. Construction of Feature Sets and method selection.

Feature selection aims to reduce the number of classification characteristics while maintaining classification accuracy [33]. After contemplating those as mentioned above dynamic and static analyses, I have decided to classify software using static analysis. In addition to its simplicity, the static analysis method has a low risk of harming mobile devices, which is one of its advantages [34].

The filter method of feature selection is a straightforward and computationally efficient technique, so it was chosen as the algorithm for feature selection in this study. The filter approach employed the Information Gain metric for attribute evaluation. This process evaluates the worth of a characteristic by assessing the amount of information gained concerning the class. This is given in the equation below;

VI.

Info Gain (Class, Attribute) = K(Class) − K (Class | Attribute) -------------------------(1)

where K is the information entropy.

E. ***Convolutional Neural Networks****:* The foundation of a convolutional neural network, also known as a CNN, is comprised of two layers: the feature mapping layer and the feature extraction layer. The task of the feature extraction layer is to establish connections between the input of each neuron and the limited receptive field that it has. Each network layer that is used for computation in the feature mapping layer is composed of a plane that contains a variety of attribute graphs. In the model that has been suggested, the nonlinear activation function known as Relu has been included into the feature mapping framework. After the images have been downsized, they are each entered into the lightweight CNN model that was advised based on the data.This simple model consists of just six layers in total: three convolutional layers, followed by three max-pooling layers, then two fully-connected layers, and finally a SoftMax output layer that is made up of C classes in their entirety. Convolutional layers increase in depth from 16 to 32 to 64 filters, with the filters of each layer measuring three by three and having a stride of one. Following each layer of convolution comes a max-pooling layer, which has a filter size of 22 dimensions and a stride value of 2. After the convolutional layers come the three Fully Connected (FC) layers, and each FC layer has 512 neural connections to the layers below it. In the final FC layer, which utilizes SoftMax activation, the C neurons represent the complete number of malware classes that are included in the training datasets. Figure 1 depicts the overall architecture of the lightweight CNN-based MDS system that is being proposed. Convolutional neural networks, often known as CNNs, typically include a pooling layer in addition to their multiple convolution layers. In order to make advantage of the qualities included in the input datasets, the convolution layer implements a nonlinear operation by dynamically adjusting the size of the window used by the convolution filtering system. The following transformation of the input I is accomplished by performing a discrete convolution with the K filter, as shown in the following formula (2)

F.

$$(IXX)_{r,s} = \sum_{u=-h_1}^{h_1} \sum_{u=-h_2}^{h_1} K_{u,v} I_{r+u,s+v} \text{.......................................}(2)$$

VII. Here the K filter is written as

VIII.

$$K = \begin{pmatrix} K_{-h1,h2} & \cdots & K_{-h1,h2} \\ \vdots & K_{0.0} & \vdots \\ K_{h1,-h2} & \cdots & K_{h1,-h2} \end{pmatrix}$$

Y_i *is* calculated as follows.

$$Y_i = \quad B_i \sum k_{i,j} \times X_j$$

IX.

G. Enhanced CNN Pseudo Code

Step 1: Start

Step 2: Introduce an initial n-dimensional dataset

Step 3: Establish Pre-processing that pulls out the null and duplicate values

Step 4: Determine Feature Selection

Step 5: Initialize CNN

Step 6: Now introduce training data into the input layer

Step 7: Initialize the SGD generalization process in the pooling layer

Step 8: Configure the Learning Rate at 0.01

Step 9: Now verify the overfitting data and perform Error loss

Step 10: When the Error loss is maximum, adjust the Learning Rate and Redo Step 7

Step 11: If Error loss is minimum, continue with the classification

Step 12: Now categorize the n dataset using RF, XGB, CNN, and ECNN

Step 13: Now forecast Precision, Accuracy and Recall

Step 14: Stop

3.1. Enhancing SVM Learning with Weighted Features

In Android Malware-related datasets, several features have been observed to be redundant or of diminished significance [35]. It is recommended to incorporate feature weights into the SVM training procedure. The benefits of rough set theory have been demonstrated in feature analysis and feature selection [36,37]. This study presents a novel method for enhancing the Support Vector Machine (SVM) algorithm. Our proposed method incorporates dimensionality reduction techniques based on Rough set theory and considers the varying levels of importance of different variables. Algorithm 1 uses rough set theory to rank features and determine their weights. Following the procedure for ranking features with a weight of 0 are deemed insignificant and are therefore eliminated. Experiments are conducted using three Android Malware datasets as part of the investigation.

```
Algorithm 1: Feature Weights Calculation
Input: Android Malware Dataset (D) with features (F)
Output: Efficient Weight Vector (WV)
Determine all of the D’s reducts through the use of rough sets;
$N_F$ ← The quantity of features in dataset D;
$N_D$ ← Represents number of reducts available in dataset D;

Step 1. Initialize the weights of each feature
for (i ← 0 to $N_D$) do
    $(WV)_i$ ← 0;
end

Step 2. Initialize the weight of each feature
    for (i ← 0 to $N_D$) do
        for (k ← 0 to $N_F$) do
            if (feature i in the kth reduct $R_k$ ) then
                n ← number of features in $R_k$ ;
                (WV)_i  ← (WV)_i  +   1/n;
              end
          end
     end
Scale the values of feature weights into the interval [0, 100]
```

3.2. Experimental Setup

Experiments were conducted in a Jupiter notebook using the programming language Python to train the Android Malware dataset. In addition, WEKA Software is utilized for data preprocessing and implementing ML components. The system is a Lenovo Core i7 CPU running Windows 10 with 32 GB RAM and 1 TB of storage space.

3.3. Dataset Details

Android Malware Data set-1: This dataset was generated with users' behaviors during malware threats. It comprehends all parameters that a user will hold in his device. The feature information and the dataset description are illustrated in the following Table 2 and Table 3.

Table -2: ANDROID MALWARE DATASET ATTRIBUTES INFORMATION

TCP_Packets	**Ist_Port_TCP**	**External_IPS**
Volume_Bytes	Udp_Packets	Tcp_urg_packet
Source_App_Packets	Remote_App_Packets	Remote_App_Bytes
Duration_Dns_Query_Times	Avg_Local_Pkt_Rate	Source_App_Packets

Table-3: ANDROID MALWARE DATASET DESCRIPTION

Dataset Characteristics:	Multivariate	**Number of Rows:**	7846	**Area:**	Security
Attribute Characteristics:	Categorical, Integer, Real	**Number of Columns:**	30	**Date Donated**	2019- 01-01
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Attributes:**	30

4. Results and Discussion

4.1. Performance Metrics

The efficacy of our approaches was tested using 10-fold cross-validation. We evaluated algorithms using accuracy, precision, recall, and F-measure. These measures are widely used in machine-learning classification [38]. Items might be true positive, false positive, false negative, or true negative.

Recall is determined from true positives and false negatives.

Recall(Rec) = T P/ (T P + F N)

Recall characterizes the proportion of predicted true outcomes among actual true outcomes.

Given the number of items classified as true positives and false positives, precision is calculated as:

Precision(Prc) = (T P + F P)/(T P).

Precision defines how much of the predicted true is actually true.

F-measure is the measurement that integrates precision and recall. F = 2 (Recall Precision) / (Precision + Recall).

Accuracy defines the overall precision of a prediction.

Accuracy(Acc)= (T P + T N) / Total Samples.

4.2. Enhanced SVM Results

The following Table 4 represents the Accuracy value of XGBoost, RF, SVM and ESVM on Android Malware Dataset -1.

Table-4 Accuracy comparison on ANDROID MALWARE DATASET -1

Algorithms	**Accuracy %**
XG-Boost	84.8
RF	91

SVM	92.1
ESVM	94

Table 4 presents the accuracy results of XG-Boost, RF, SVM, and ESVM algorithms on the Android Malware Dataset-1. Notably, the enhanced version of SVM exhibits the highest accuracy among all the other methods.

The following Fig. 4 shows the graphical format of the accuracy value comparison of XGBoost, RF, SVM, and ESVM on the Android Malware Dataset-1.

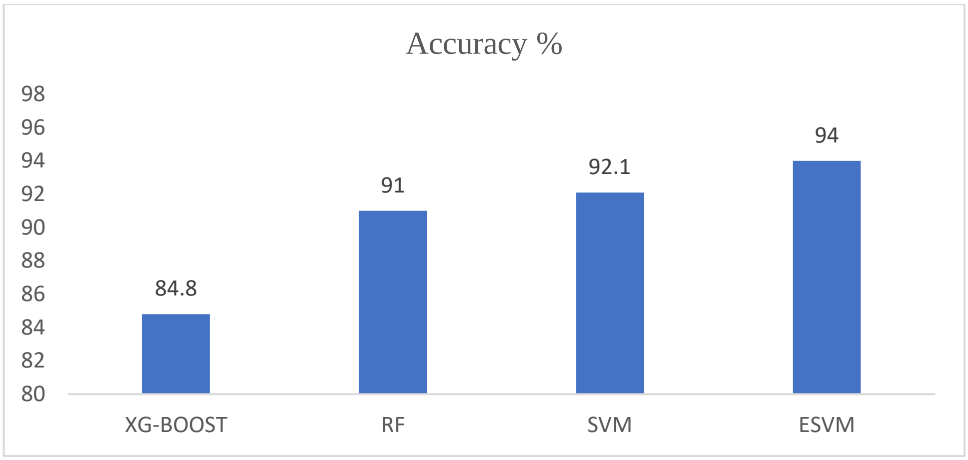

Fig. 4. Accuracy value of XGBoost, RF, SVM and ESVM on Android Malware Dataset-1

The Enhanced SVM (94%) has the highest level of accuracy, followed by the SVM (92.2%), the RF (91%), and the XGB (84.8%). Table 4 presents the accuracy results of XGBoost, RF, SVM, and ESVM algorithms on the Android Malware Dataset-1. Notably, the enhanced version of SVM exhibits the highest accuracy among all the other methods.

The following Fig.5 shows the graphical format of the precision and recall value comparison of XGBoost, RF, SVM, and ESVM on Android Malware Dataset-1. The graph clearly shows that ESVM has the best precision and recall, followed by SVM, RF, and XGB. Both the SVM and the ESVM have high precision values relative to their recall. However, the precision values perform better than the recall values in RF and XGB.

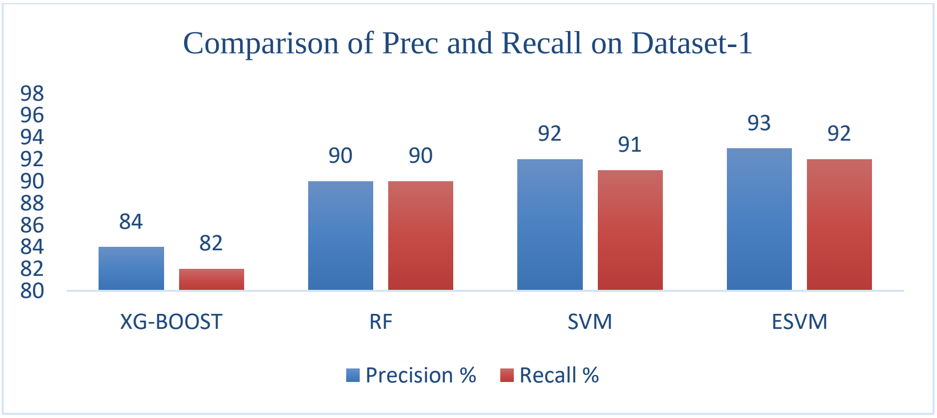

Fig. 5. Precision and Recall value Comparisons of XGBoost, RF, SVM and ESVM on Android Malware Dataset -1

Table 5 displays the precision and recall values of XGBoost, RF, SVM, and ESVM on the first Android Malware Dataset. SVM (92%, 91%) and ESVM (93%, 92%) demonstrate high precision and recall.

Table 5: Comparison of performance metrics Prec. and Rec. on ANDROID MALWARE DATASET-1

Algorithms	**Precision %**	**Recall%**
XG-Boost	84	82
RF	90	90
SVM	92	91
ESVM	93	92

Fig.5 displays the precision and recall values of XGBoost, RF, SVM, and ESVM on the first Android Malware Dataset-1. Both the SVM and the ESVM have high precision values relative to their recall. However, the precision values perform better than the recall values in RF and XGB.

Table 6: Accuracy comparison on ANDROID MALWARE DATASET-1

Algorithms	Accuracy %
XG-Boost	84.8
RF	91
CNN	92.6
ECNN	95.8

Table 6 presents the accuracy results of XGBoost, RF, CNN, and ECNN algorithms on the Android Malware Dataset-1. Notably, the enhanced version of ECNN exhibits the highest accuracy among all the other methods.

The following Fig. 6 shows the graphical format of the accuracy value comparison of XGBoost, RF, CNN, and ECNN on Android Malware Dataset-1.

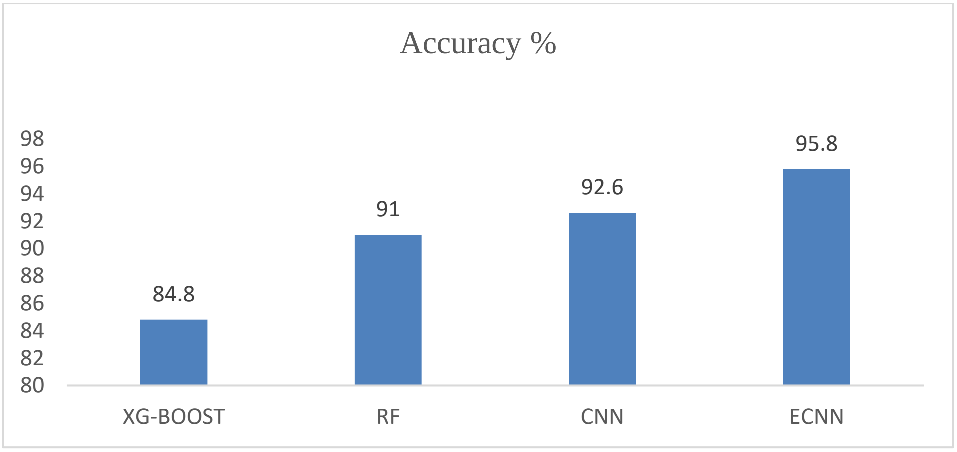

Fig. 6. Accuracy value Comparisons of XGBoost, RF, CNN and ECNN on Android Malware Dataset -1

The Following Table 7 shows the Precision(Prc.) and Recall(Rec.) value of XGBoost, RF, CNN, and ECNN on Android Malware Dataset-1 and its graphical presentation is depicted in Fig-7. Both the ECNN and the CNN have high precision values relative to their recall. However, the precision values perform better than the recall values in CNN and ECNN.

Table 7: Performance metrics comparison on ANDROID MALWARE DATASET -1

Algorithms	Precision%	Recall%
XG-Boost	84	82
RF	90	90
CNN	94	91
ECNN	96	92

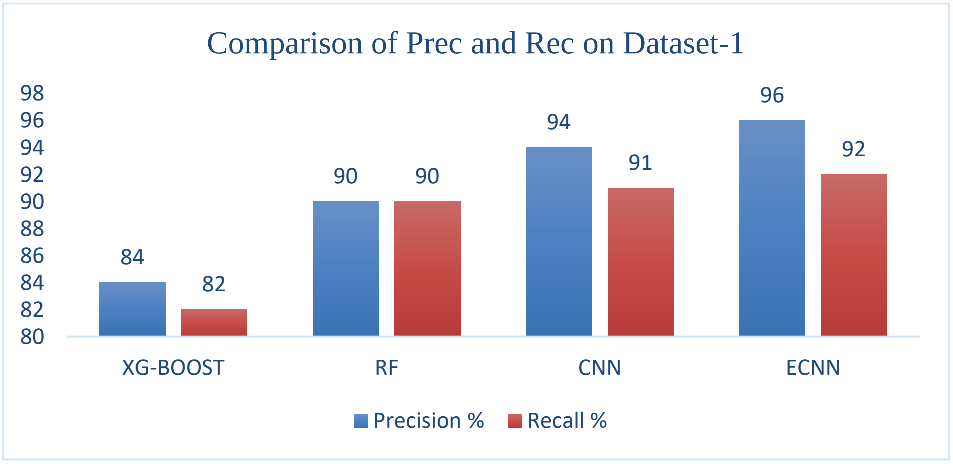

Fig. 7. Prec. and Rec. comparison of XGB, RF, CNN and ECNN on Android Malware Dataset -1

4.3. Dataset: Android Malware Dataset -2

Data Set Information: This dataset has been taken from a private bank. The Android Malware Dataset -2 is a cyber hacking dataset. This database contains 23 attributes with 4425 rows and 23 columns.. It consists of all details of customers who have been affected by cyber hacking breaches in a particular year. The following Tables 8 and 9 describe the attributes information and dataset information.

Table 8: ATTRIBUTES INFORMATION OF KAGGLE ANDROID MALWARE DATASET -2

Serial Number	1 to 4425	**Customer ID**	1 to 5425
Business Associate	Type of Business	**Individuals Affected**	No of Individuals
Type of Breach	Hacking/ Theft	**Breach Location**	Location
Summary Year	Affected/ Others Year of breach	**Breach Start**	Starting Time

Table 9: DATASET INFORMATION OF ANDROID MALWARE DATASET-2

Data Set Characteristics	Multivariate	**Number of Rows:**	4425	**Area:**	Finance
Attribute Characteristics:	Categorical, Integer, Real	**Number of Columns:**	23	**Date Donated**	2019-01-01
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Attributes:**	23

4.3.1. Result Analysis on Android Malware Dataset-2

The following section has the analysis of the obtained results on Android Malware Dataset-2.

Table 10 shows the accuracy comparison of XGB, RF, SVM, and ESVM, while Table 11 presents the comparison of precision and recall value of XGB, RF, SVM, and ESVM on Android Malware Dataset-2. Fig. 8 is used to represent the graphical format of accuracy comparison of XGB, RF, SVM, and ESVM on Android Malware Kaggle Dataset-2. The following Fig. 9 shows the graphical format of the precision and recall value comparison of XG Boost, RF, SVM, and ESVM on Android Malware Dataset-2. The precision values in each algorithm perform better than the recall values. ESVM provides the best result, followed by SVM, RF, and XGB. Each method performs better when precision settings are used than recall values. The best approach is offered by ESVM, followed by SVM, RF, and XGB.

Table 10: Accuracy on ANDROID MALWARE DATASET-2

Algorithms	**Accuracy%**
XG- Boost	87.41
RFt	91.75
SVM	92.91
ESVM	93.97

Table 11:Performance metrics on ANDROID MALWARE DATASET-2

Algorithms	**Precision %**	**Recall %**
XG-Boost	87	82
RF	90	87
SVM	91	88
ESVM	93	91

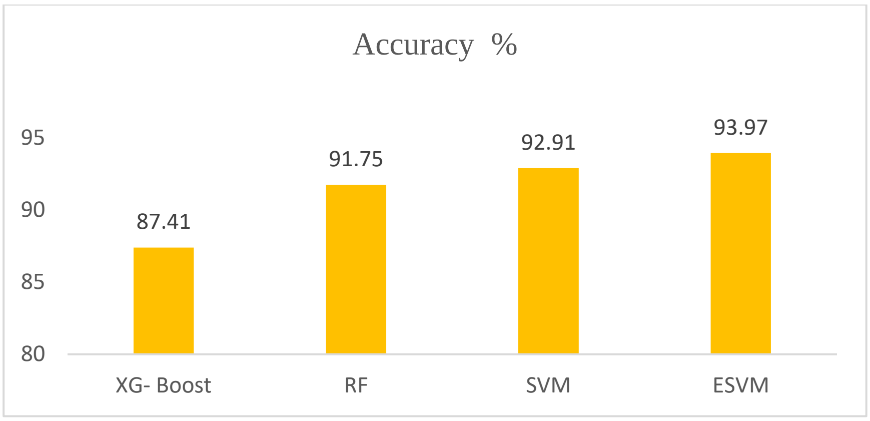

Fig. 8. Accuracy value Comparisons of XGBoost, RF, SVM and ESVM on Android Malware Dataset-2

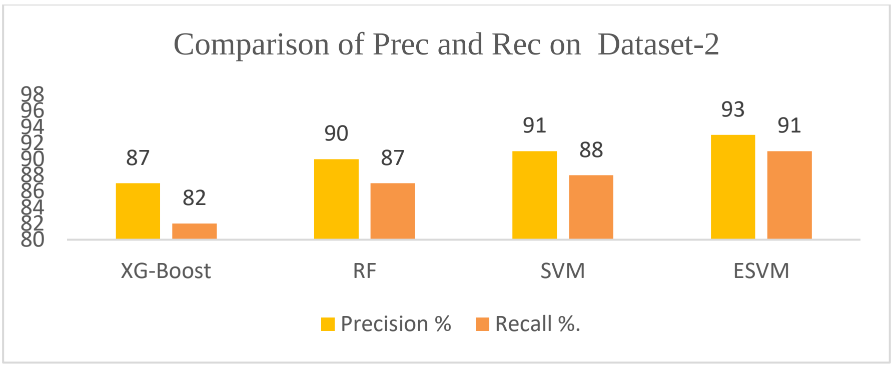

Fig.9. Prec. and Rec. value comparisons of XG-Boost, RF, SVM and ESVM on Android Malware Dataset -2

Table 12 compares the experimentally determined accuracy of XGB, RF, CNN, and ECNN. Table 13 contrasts this by showing how XGB, RF, CNN, and ECNN do on Android Malware Dataset-2 in terms of accuracy and recall. The accuracy comparison of XGB, RF, CNN, and ECNN is shown graphically in Fig. 10. The comparison of XG Boost, RF, CNN, and ECNN on Android Malware Dataset-2 in terms of accuracy and recall values is shown graphically in Fig. 11. When compared to other CNNs, the ECNN performed the best. In all algorithms, the recall values are lower than the precision values. When comparing accuracy, ECNN comes out on top, followed by CNN, RF, and XGB.

Table 12:ACCURACY VALUE on ANDROID MALWARE DATASET -2

Algorithms	**Accuracy %**
XG-Boost	87.41
RF	91.75
CNN	95.5
ECNN	96.14

Table 13: Performance metrics on ANDROID MALWARE DATASET-2

Algorithms	**Precision %**	**Recall %**
XG-Boost	87	82
RF	90	87
CNN	94	92
ECNN	94	92

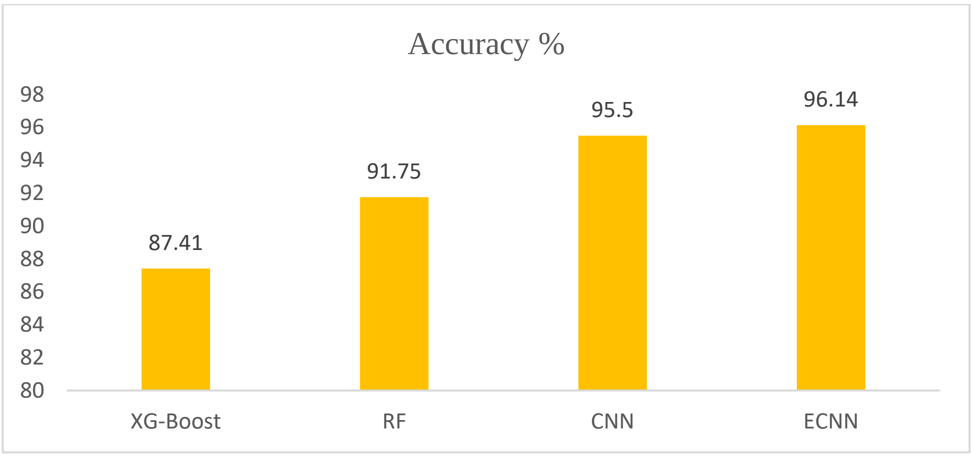

Fig. 10. Accuracy on Android Malware Dataset-2

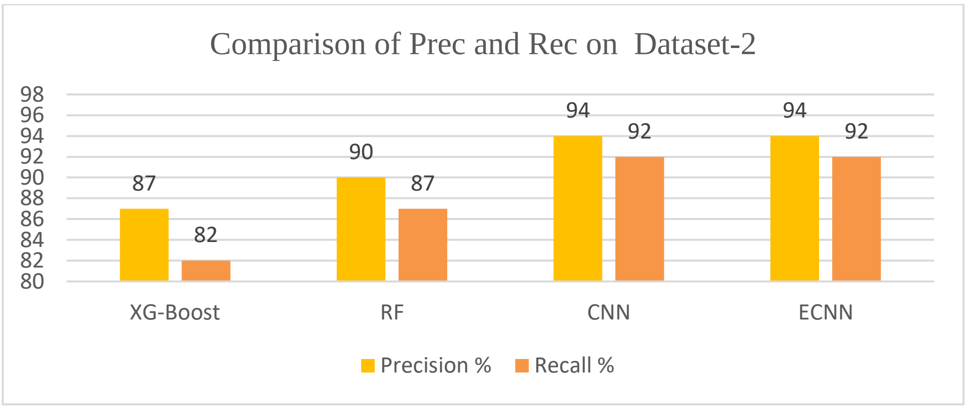

Fig. 11. Performance on Android Malware Dataset -2

4.4. Dataset: Android Malware Dataset -3

Data Set Information: The dataset contains the details of Android malware attack which is been happened over an android network. This database contains 20 features with 5850 instances. Following Table 14 and 15 describes the attributes information and dataset information.

Table 14: ATTRIBUTES INFORMATION OF ANDROID MALWARE DATASET-3

Source Port	Source IP	**Destination Port**	Destination IP	**NAT Source Port**	Source IP
NAT Destination Port	Destination IP	**Action**	Allow / Don't Allow	**Bytes**	50 to 50000
Byte Sent	50 to 50000	**Bytes Received**	50 to 50000	**Packets**	50 to 500
Elapsed Time	2 to 1500	**Packet Sent**	50 to 5000	**Packet Received**	50 to 5000

Table 15: DATASET INFORMATION OF ANDROID MALWARE DATASET-3

Data Set Characteristics	Multivariate	**Number of Rows:**	5850	**Area:**	Computer
Attribute Characteristics:	Categorical, Integer, Real	**Number of Columns:**	20	**Date Donated**	2020-05-07
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Attributes:**	20

Table 16: Accuracy on ANDROID MALWARE DATASET -3

Algorithms	Accuracy %
XG-Boost	89.19
RF	92.68
SVM	93.11
ESVM	95.46

Table 17: Performance metrics on ANDROID MALWARE DATASET -3

Algorithms	Precision %	Recall %
XG-Boost	89	83
RF	90	89
SVM	93	92
ESVM	95	93

The Table 16 represents Accuracy value of XGBoost, RF, SVM and ESVM on Android Malware Dataset-3. It is evident from the table ESVM obtained comparatively better result among others.

Table 17 represents the Precision and Recall comparison obtained in the experiment by XG-Boost, RF, SVM, and ESVM on Android Malware Dataset-3. It is evident from the table that ESVM received the best result among others in terms of Precision and Recall.

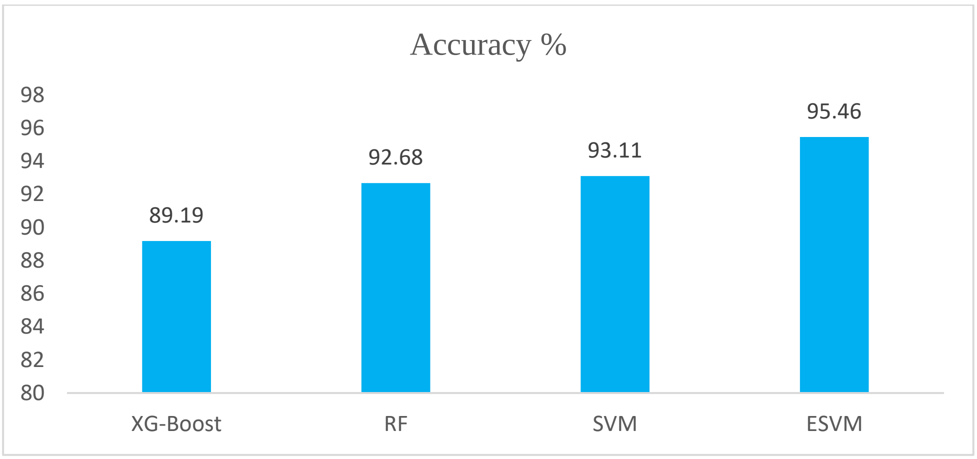

Fig. 12. Accuracy on Android Malware Dataset-3

Fig. 12 shows the graphical format of accuracy comparison of XGBoost, RF, SVM, and ESVM on Android Malware Dataset-3. Fig. 13 represents the Precision and Recall value comparisons of XG-Boost, RF, SVM and ESVM on Android Malware Dataset-3.

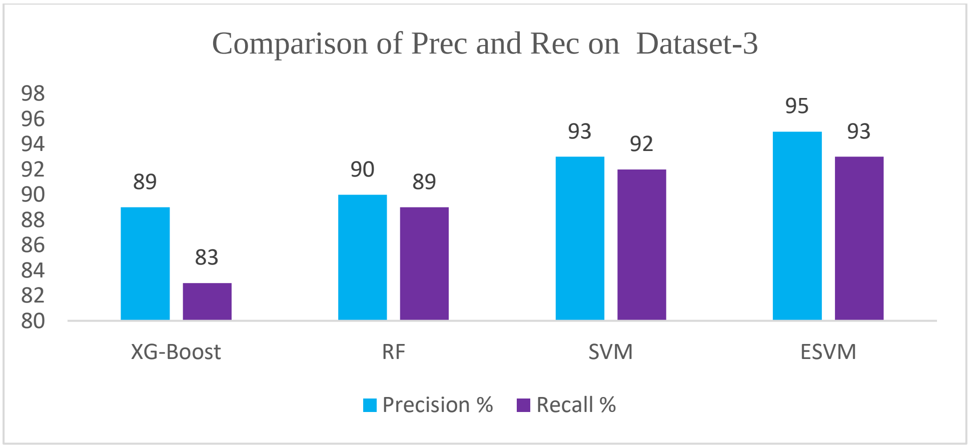

Fig. 13. Performance metrics on Android Malware Dataset-3

4.4.1. Enhanced CNN Results:

Table 18: ACCURACY on ANDROID MALWARE DATASET -3

Algorithms	Accuracy %
XG-Boost	89.19
RF	92.68
CNN	96.88
ECNN	96.92

Table 19: Performance metrics on ANDROID MALWARE DATASET -3

Algorithms	Precision %	Recall %
XG-Boost	89	83
RF	90	89
CNN	96	94
ECNN	96	94

Table 18 represents the Accuracy value of XGBoost, RF, CNN, and ECNN on Android Malware Dataset-3. ECNN achieved the best result, 92.3%, in comparison to others, while XGB showed the lowest accuracy value at 82%. Table 19 shows the precision and recall value of XGBoost, RF, CNN, and ECNN on Android Malware Dataset-3. In terms of both the precision and recall value, ECNN showed the best performance among others.

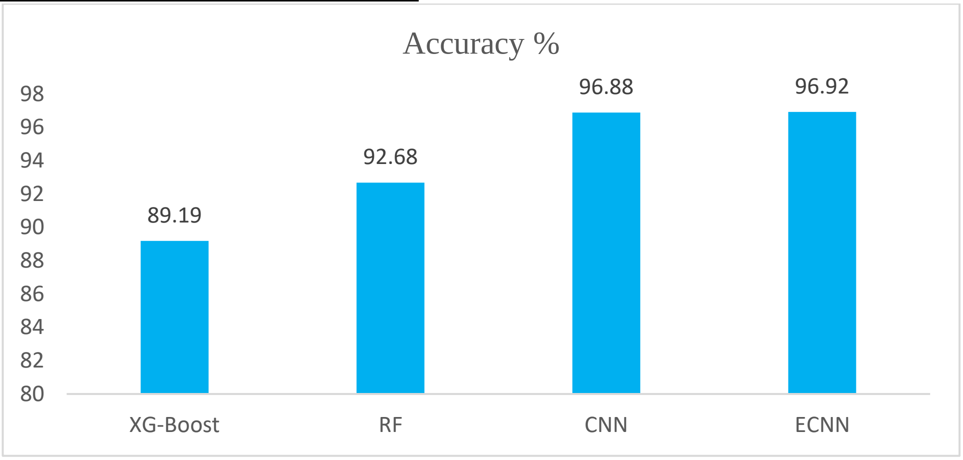

Fig. 14. Accuracy on Android Malware Dataset-3

Fig. 14 shows the graphical format of accuracy comparison of XGBoost, RF, CNN and ECNN on Android Malware Dataset-3. Fig. 15 represents the Precision and Recall value comparisons of XG-Boost, RF, CNN and ECNN on Android Malware Dataset-3.

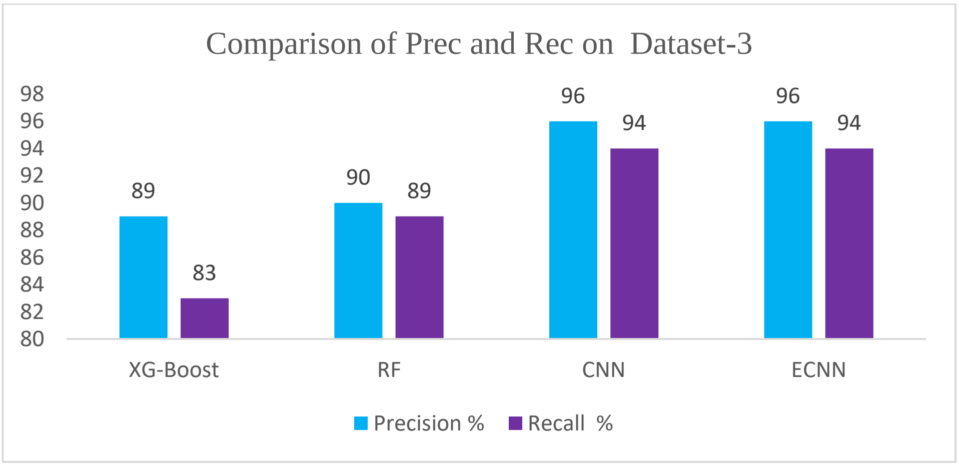

Fig. 15. Performance metrics on Android Malware Dataset-3

Table 20 Performance Metrics on ANDROID MALWARE DATASET -1, ANDROID MALWARE DATASET -2 AND ANDROID MALWARE DATASET -3

Dataset Name	**Parameters**	**X Gradient Boost**	**Random Forest**	**Support Vector Machine**	**Enhanced SVM**	**Convolution Neural Network**	**Enhanced Convolution Neural Network**
		(XG-Boost)	**(RF)**	**(SVM)**	**(ESVM)**	**(CNN)**	**(ECNN)**
Android Malware Dataset-1	**Accuracy %**	84.8	91	92.1	94	92.6	95.8
	Precision %	0.84	90	92	93	94	96
	Recall%	0.82	90	91	92	91	92
Android Malware Dataset-2	**Accuracy %**	87.41	91.75	92.91	93.97	95.05	96.14
	Precision %	87	90	91	93	94	94
	Recall %	82	87	88	91	82	92
Android Malware Dataset-3	**Accuracy %**	89.19	92.68	93.11	95.46	96.88	96.92
	Precision %	89	90	93	95	96	96
	Recall %	83	89	92	93	94	94

Table 20 displays the accuracy, precision, and recall value of various machine learning models, namely XGBoost, RF, CNN, ECNN, SVM, and ESVM, on three distinct datasets, namely Android Malware Dataset-1, Android Malware Dataset-2, and Android Malware Dataset-3.

This table shows the summarized results of the proposed CNN-based Ensemble approach with the different datasets. In an Android Malware Dataset-1, the ECNN (95.8%) accuracy outperforms all the other methods. Whereas ESVM (94 %) performs the second best, followed by CNN(92.6%), SVM (92.1%), RF (91%), and XGB (84.8%).

ECNN and CNN (96%, 94%) performed fine in precision-based values among all other methods. However, the EVSM (93%) achievement approaches the ECNN and CNN. The least precision value is for XGB (84%). The highest recall values are for ECNN and ESVM (92%), and the lowest is for XGB (82%).

The ECNN (96.14%) accuracy beats all other approaches in an Android Malware Dataset-2. The CNN (95.05%) comes in second, followed by the ESVM (93.97%), SVM (92.91%), RF (91.75%), and XGB (87.41%). Regarding precision, both ECNN and CNN (94%) performed well compared to all other values. However, the EVSM (93%) approaches the ECNN and CNN and is followed by the SVM (91%), RF (90%), and XBG (87%). The recall values for ECNN and CNN were the same, 92%, followed by ESVM (91%), SVM (88%), RF (87%), and XGB (82%).

The ECNN (96.92%) accuracy exceeds all other approaches in an Android Malware Dataset-3. CNN (96.88%) comes in second, followed by ESVM (95.46%), SVM (93.11%), RF (92.68%), and XGB (89.19%). ECNN and CNN (96%) performed well in precision-based values compared to all other values. The EVSM (95%), on the other hand, performs very close to the ECNN and CNN, followed by the SVM (93%), RF (90%), and XBG (89%). The recall values for ECNN and CNN were the same, 94%, followed by ESVM (93%), SVM (92%), RF (89%), and XGB (83%).

5. Conclusion

According to Omar[39], traditional AI approaches, specifically ML algorithms, are no longer effective in identifying all complex variant-type malware. DL approach, which is fairly different from ML concepts, can be a promising explanation for the difficulty of identifying all malware variants. The experimental results obtained during the implementation of Android malware analysis with static analysis technique prove that the permission-related algorithm works effectively in DL and ML methods. Though the accuracy of the above-discussed techniques is very close to each other, few differences exist in the running time required by the algorithms. The observed result demonstrates that the ECNN method works with the BP (Back Propagation) model for every layer between multiple intermediate layers, rendering a much faster and higher rate of accuracy when compared to other algorithms. Additionally, ML algorithms were classified by employing 80% training and 20% test data, as shown in Fig.16, in the same way as ECNN. Hence, the quantity of data utilized in the study is excessive, and the dataset contains only two aims, including benign and malicious, where it is noted that the accuracy rate attained is 96%.

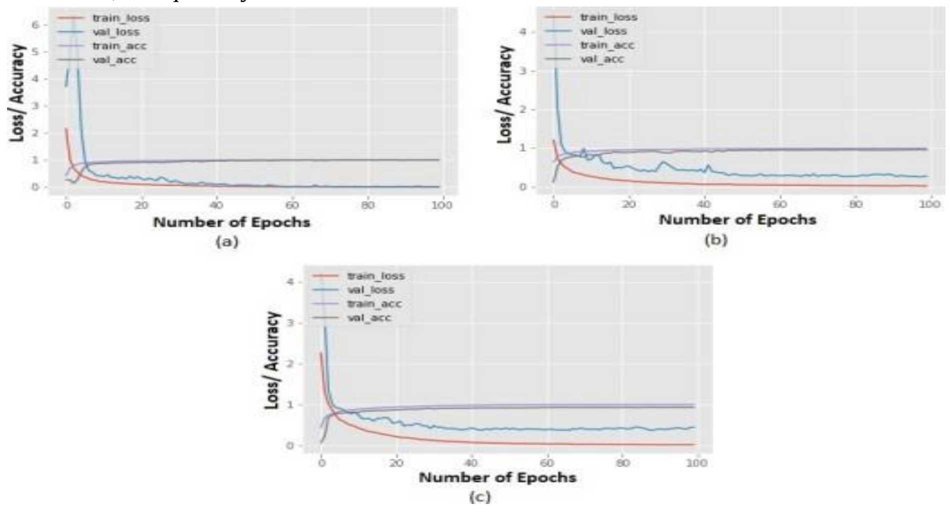

Fig. 16. Plot showing training and validation accuracy and loss for Lightweight CNN- based MDS with the three dataset.

References

[1] S. K. Sasidharan and C. Thomas, “Prodroid-an an droid malware detection framework based on profile hid- den markov model,” *Pervasive and Mobile Computing*, vol. 72, pp. 101 336–101 336, 2021.

[2] L. N. Vu and S. Jung, “Admat: A cnn-on-matrix approach to android malware detection and classification,” *IEEE Access*, vol. 9, pp. 39–680, 2021.

[3] Thakkar, A.; Lohiya, R. A Review on Machine Learning and Deep Learning Perspectives of IDS for IoT: Recent Updates, Security Issues, and Challenges; Springer: Dordrecht, The Netherlands, 2021; Volume 28, pp. 3211–3243. [CrossRef] 10.1007/s11831-020-09496-0

[4] Gowdhaman, V.; Dhanapal, R. An intrusion detection system for wireless sensor networks using deep neural network. Soft Comput. 2021, 26, 13059–13067. 10.1007/s00500-021-06473-y

[5] Liu, K.; Xu, S.; Xu, G.; Zhang, M.; Sun, D.; Liu, H. A review of android malware detection approaches based on machine learning. IEEE Access 2020, 8, 124579–124607. 10.1109/ACCESS.2020.3006143

[6] Bovenzi, G.; Persico, V.; Pescapé, A.; Piscitelli, A.; Spadari, V. Hierarchical Classification of Android Malware Traffic. In Proceedings of the 2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Wuhan, China, 9–11 December 2022; pp. 1354–1359. 10.1109/TrustCom56396.2022.00191

[7] D. O. S¸ahın, S. Akleylek, and E. Kili¸c, “Linregdroid: Detection of android malware us- ing multiple linear re- gression models-based classifiers,” *IEEE Access*, vol. 10, pp. 14–246, 2022.

[8] Y. Hei, R. Yang, H. Peng, L. Wang, X. Xu, J. Liu, H. Liu, J. Xu, and L. Sun, “Hawk: Rapid android malware detection through heterogeneous graph attention networks,” *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[9] S. S. Vanjire and M. Lakshmi, “Mdta: A new approach of supervised machine learning for android malware detection and threat attribution using behavioral reports,” *Mobile Computing and Sustainable Informatics*, pp. 147– 159, 2022.

[10] Karbab, E.M.B.; Debbabi, M.; Derhab, A.; Mouheb, D. MalDozer: Automatic framework for android malware detection using deep learning. Digit. Investig. 2018, 24, S48–S59. 10.1016/j.diin.2018.01.007

[11] S.Y. Yerima, S. Khan, Longitudinal performance analysis of machine learning based Android malware detectors, in: 2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), IEEE, 2019, pp. 1–8.

[12] J. Garcia, M. Hammad, S. Malek, Lightweight, obfuscation-resilient detection and family identification of android malware, ACM Trans. Software Eng. Methodol. 26 (3) (2018) 1–29

[13] K.G. Kumar, Efficient android malware scanner using hybrid analysis, Int. J. Recent Technol. Eng. 7 (2019) 76–80

[14] D. Teubert, J. Krude, S. Schueppen, U. Meyer, Hugin:

a scalable hybrid android malware detection system, in: SECURWARE 2017: the Eleventh International Conference on Emerging Security Information, Systems and Technologies, 2017, pp. 168–176.

[15] N. Daoudi, J. Samhi, A. K. Kabore, K. Allix, T. F. B. e, and J. Klein, “Dexray: A simple, yet effective deep learning approach to android malware detection based on image representation of bytecode,” *International Work- shop on Deployable Machine Learning*

[16] H.Bai, N. Xie, X. Di, and Q. Ye, “Famd: A fast multifeature android malware detection framework, design and implementation,” *IEEE Access*, vol. 8, pp. 194–729, 2020

[17] A. Mehtab, W. B. Shahid, T. Yaqoob, M. F. Amjad, H. Abbas, H. Afzal, and M. N. Saqib, “Addroid: rule-based machine learning framework for android malware analysis,” *Mobile Networks and Applications*, vol. 25, no. 1, pp. 180–192, 2020.

[18] L. Cai, Y. Li, Z. Xiong, and Jowmdroid, “Android mal- ware detection based on feature weighting with joint op- timization of weight-mapping and classifier parameters,” *Computers & Security*, vol. 100, pp. 102 086–102 086, 2021.

[19] S. I. Imtiaz, S. U. Rehman, A. R. Javed, Z. Jalil, X. Liu, and W. S. Alnumay, “Deepamd: Detection and identi- fication of android malware using high-efficient deep artificial neural network,” *Future Generation computer systems*, vol. 115, pp. 844–856, 2021.

[20] W. Zhang, H. Wang, H. He, and P. Liu, “Damba: detect- ing android malware by orgb analysis,” *IEEE Transac- tions on Reliability*, vol. 69, no. 1, pp. 55–69, 2020.

[21] H. Gao, S. Cheng, and W. Zhang, “Gdroid: Android malware detection and classification with graph convo- lutional network,” *Computers & Security*, vol. 106, pp. 102 264–102 264, 2021.

[22] A. Guerra-Manzanares, H. Bahsi, and S. N. omm, “Kron-odroid Time-based hybrid- featured dataset for effective android malware detection and characterization,” Com- puters & Security, vol. 110, pp. 102 399–102 399, 2021.

[23] Y. Ban, S. Lee, D. Song, H. Cho, and J. H. Yi, “Fam: Featuring android malware for deep learning-based familial analysis,” IEEE Access, vol. 10, pp. 20–28, 2022.

[24] H. Cai and B. G. Ryder, “Droidfax: A toolkit for system- atic characterization of android applications,” 2017 IEEE International Conference on Software Maintenance and Evo- lution (ICSME), pp. 643–647, 2017.

[25] E. Mariconti, L. Onwuzurike, P. Andriotis, E. D. Cristo- faro, G. Ross, G. Stringh-Ini, and . . Mamadroid, 2016.

[26] K. Xu, Y. Li, R. Deng, K. Chen, and J. Xu, “Droide-volver: Self-evolving android mal- ware detection system,” 2019 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 47–62, 2019.

[27] W. Li, X. Fu, and H. Cai, “Androct: Ten years of app call traces in android,” 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), pp. 570–574, 2021.

[28] N. R. Surendran, T. Thomas, and S. Emmanuel, “Gsdroid: Graph signal based compact feature representation for android malware detection,” Expert Systems with Appli- cations, vol. 159, pp. 113 581–113 581,2020.

[29] E. B. Karbab and M. Debbabi, “Petadroid: Adaptive android malware detection using deep learning,” International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 319–340, 2021.

[30] Zhao, X.; Fang, J.; Wang, X. Android malware detection based on permissions. In Proceedings of the ICICT 2014, Nanjing, China, 2 October 2014. 10.1049/cp.2014.0605

[31] Emanuelsson, P.; Nilsson, U. A comparative study of industrial static analysis tools. Electron. Notes Theor. Comput. Sci. 2008, 217, 5–21. 10.1016/j.entcs.2008.06.039

[32] 15. Wang, W.; Zhao, M.; Wang, J. Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network. J. Ambient. Intell. Humaniz. Comput. 2019, 10, 3035–3043. 10.1007/s12652-018-0803-6

[33] Raymer, M.L.; Punch, W.F.; Goodman, E.D.; Kuhn Leslie, A.; Jain, A.K. Dimensionality reduction using genetic algorithms. IEEE Trans. Evol. Comput. 2000, 4, 164–171. 10.1109/4235.850656

[34] Bhattacharya, A.; Goswami, R.T.; Mukherjee, K. A feature selection technique based on rough set and improvised PSO algorithm (PSORS-FS) for permission based detection of Android malwares. Int. J. Mach. Learn. Cybern. 2018, 10, 1893–1907. 10.1007/s13042-018-0838-1

[35] Han, J.C., Sanchez, R., Hu, X.H.,: Feature Selection Based on Relative Attribute Dependency: An Experimental Study. RSFDGrC’05, I, LNAI. 3641 (2005) 214-223.

[36] Hu, K., Lu, Y., Shi, C.: Feature Ranking in Rough Sets. AI Communications. 16 (2003) 41-50

[37] Yao, J.T., Zhang, M.: Feature Selection with Adjustable Criteria. RSFDGrC'05, I, LNAI. 3641 (2005) 204–213.

[38] Boiy, M.-F. Moens, A machine learning approach to sentiment analysis in multilingual web texts. Information retrieval 12(5), 526–558 (2009)

[39] Ö. Aslan and A. A. Yilmaz, "A New Malware Classification Framework Based on Deep Learning Algorithms," in IEEE Access, vol. 9, pp. 87936-87951, 2021, doi: 10.1109/ACCESS.2021.3089586