IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 28 December 2023, accepted 21 January 2024, date of publication 24 January 2024, date of current version 20 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3357944

RESEARCH ARTICLE

Intelligent Pattern Recognition Using Equilibrium Optimizer With Deep Learning Model for Android Malware Detection

MOHAMMED MARAY[1], MASHAEL MAASHI[2], HAYA MESFER ALSHAHRANI[3], SUMAYH S. ALJAMEEL[4], SITELBANAT ABDELBAGI[5], AND AHMED S. SALAMA[6]

[1]Department of Information Systems, College of Computer Science, King Khalid University, Abha, Saudi Arabia
[2]Department of Software Engineering, College of Computer and Information Sciences, King Saud University, P.O. Box 103786, Riyadh 11543, Saudi Arabia
[3]Department of Information Systems, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia
[4]Saudi Aramco Cybersecurity Chair, Department of Computer Science, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, Dammam 31441, Saudi Arabia
[5]Department of Computer and Self Development, Preparatory Year Deanship, Prince Sattam bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia
[6]Department of Electrical Engineering, Faculty of Engineering and Technology, Future University in Egypt, New Cairo 11845, Egypt

Corresponding author: Sitelbanat Abdelbagi (s.abdelbagi@psau.edu.sa)

The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University for funding this work through large group Research Project under grant number (RGP2/248/44). Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2024R237), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. Research Supporting Project number (RSPD2024R787), King Saud University, Riyadh, Saudi Arabia. This study is supported via funding from Prince Sattam bin Abdulaziz University project number (PSAU/2023/R/1444). This study is partially funded by the Future University in Egypt (FUE).

ABSTRACT Android malware recognition is the procedure of mitigating and identifying malicious software (malware) planned to target Android operating systems (OS) that are extremely utilized in smartphones and tablets. As the Android ecosystem endures to produce, therefore is the risk of malware attacks on these devices. Identifying Android malware is vital for keeping user data, privacy, and device integrity. Android malware detection utilizing deep learning (DL) signifies a cutting-edge system for the maintenance of mobile devices. DL approaches namely recurrent neural network (RNN) and convolutional neural network (CNN) are best in automatically removing intricate designs and behaviors in Android app data. By leveraging features such as application programming interface (API) call sequences, code patterns, and permissions, these approaches are efficiently differentiated between benign and malicious apps, even in the face of previous unseen attacks. This study presents an Intelligent Pattern Recognition using an Equilibrium Optimizer with Deep Learning (IPR-EODL) Approach for Android Malware Recognition. The purpose of the IPR-EODL approach is to properly identify and categorize the Android malware in such a way that security can be achieved. In the IPR-EODL technique, the data pre-processing step was applied to convert input data into a compatible setup. In addition, the IPR-EODL technique applies channel attention long short-term memory (CA-LSTM) methodology for the recognition of Android malware. To enhance the solution of the CA-LSTM algorithm, the IPR-EODL system employs the Equilibrium optimization (EO) algorithm for the hyperparameter tuning method. The experimentation evaluation of the IPR-EODL model can be verified on a benchmark Android malware database. The extensive results highlight the significant result of the IPR-EODL approach to the Android malware detection process.

INDEX TERMS Artificial intelligence, pattern recognition, android malware, security, deep learning.

I. INTRODUCTION

In the present scenario, Cybersecurity has become a primary area for computer scientists and network engineers that provide satisfactory solutions to numerous problems [1]. As the outcome of the fast developments in technological growth and their essential integrations in every form of our lives, a miscellany of malware features and their expected goals have become recognized as well as studied [2]. Among the malware diversity, the Android malware has received

The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Mueen Uddin.

attention lately which is found to occupy considerable attention globally. Android is a common operating system (OS) when compared to other that controls the OS market. Malware attacks are familiar situations that can be recognized as an attack on Android [3]. Numerous descriptions for malware were proposed by various researchers that depend on the damages caused. The decisive significance of the malware lies in malicious apps using different forms of encryption with the aim of gaining unauthorized access. It is aimed to perform actions that are neither authorized nor appropriate during a security breach. The three core regulations in security are namely: privacy, availability, and integrity [4].

Malware associated with smart systems can be recognized in three contexts as threats such as objectives & attitude, sharing and infection routes, and privilege acquisition modes. Scams, junk e-mails, information thievery, and abuse of websites can be stated as hazardous aims and behavioral outlooks. Software, browsers, markets, systems, and networks can be recognized as the sharing and infected directions [5]. Procedural exploitation and employer management like social engineering are enumerated in the privilege and acquisition means. Malware especially associated with an Android OS has been recognized as Android malware that damages or takes information in an Android-based mobile device [6]. These are considered ransomware, adware, botnet, spyware, worms, backdoors, and Trojans.

Google defines malware as possibly malicious features [7]. They categorized malware as commercial and non-commercial privilege escalation, spyware, phishing, and kinds of frauds like Trojans, backdoors toll fraud, and SMS fraud. To reduce the threat via Android malware, several researchers approached and developed it so far. The malware identification methods are divided into two types namely dynamic and static analysis-based classification [8]. Dynamic analysis has the benefit that it is likely to control malicious features that employ complicated methods like code packing or encryption. Both machine learning (ML) and DL methods have shown effectiveness in the malware detection process, especially in the context of Android feature analysis and broader areas of cybersecurity [9]. DL methods are found to be highly efficient over traditional ML methods for Android malware recognition. Static analysis, based on algorithms, applies linguistic features that can be eliminated with no implementation of a feature, while dynamic analysis-based methods utilize semantic features observed when an application is executed in specific environments [10].

ML and non-ML approaches grapple with problems like restricted efficiency in developing malware patterns and a superior incidence of false positives. Afterward, DL approaches were determined useful, highlighting their ability to automatically extract intricate features and recognize difficult patterns, preparing them more capable of tackling the dynamic nature of Android malware. By concentrating on this feature, the research endeavors to improve the performance of Android malware detection, assisting in more correct and effective solutions in the ever-evolving landscape of cybersecurity attacks.

This study presents an Intelligent Pattern Recognition using an Equilibrium Optimizer with Deep Learning (IPR-EODL) Approach for Android Malware Recognition. The main purpose of the IPR-EODL system is to properly identify and categorize Android malware in such a way that security can be achieved. In the IPR-EODL technique, the data preprocessing step was executed to change input data into a compatible setup. In addition, the IPR-EODL methodology applies channel attention long short-term memory (CA-LSTM) algorithm for the recognition of Android malware. To enhance the solution of the CA-LSTM algorithm, the IPR-EODL approach employs the EO algorithm for the hyperparameter tuning procedure. The experimentation evaluation of the IPR-EODL methodology can be tested on a benchmark Android malware database. The key contributions of the study are listed as follows.

- Establish an IPR-EODL algorithm for Android malware detection, presenting a new technique to address the problems in this field. The main objective of the IPR-EODL algorithm is to accurately detect and classify Android malware, demonstrating the impact of robust security measures.
- Utilizes CA-LSTM approach for efficient analysis and detection of Android malware. It depicts conventional approaches by capturing long-term dependencies and concentrating on important features with channel attention.
- Combines the EO method for optimizing hyperparameters of the CA-LSTM algorithm. This optimizer models efficiency and potentially boosts accuracy related to fixed or manually tuned parameters.

II. LITERATURE REVIEW

The authors in [11] proposed an intelligent hyperparameter-altered DL-based malware detection (IHPT-DLMD) approach. Also, the Bi-directional Long Short-Term Memory (BiLSTM) methodology has been deployed for the classification and recognition of Android malware. Finally, the glowworm swarm optimization (GSO) approach has been presented to enhance the hyperparameters of the BiLSTM method. The proposed model is assessed by a standard dataset. The authors in [12] presented an ML-based system for Android malware recognition. Also, the devised technique makes use of the Information Gain model. They make use of various integrations of API Calls, contextual features, and authorizations. These integrations were fed into different ML models. The authors in [13] presented a new technique for recognizing malware in Android apps by GRU, which is a type of RNN technique.

In [10], an ML-based recognition technique that makes use of hybrid analysis-based Particle swarm optimization (PSO) and an adaptive genetic algorithm (AGA) is proposed. Primarily, feature selection (FS) was done by employing PSO to the features. Then, the demonstration of XGBoost and

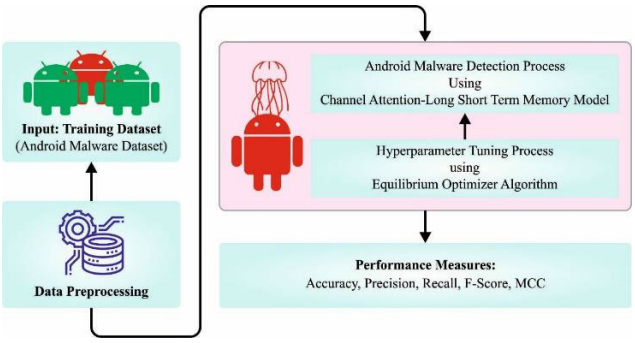

FIGURE 1. The overall flow of IPR-EODL algorithm.

random forest (RF) using ML identifiers is enhanced by the AGA. The author in [14] projected a structure employing image-based malware depictions of Android Dalvik Executable (DEX) files. The structure utilizes the EffectiveNetB0 convolutional neural network (CNN) for feature abstraction, which was then delivered over a worldwide average pooling layer and provided in a stacking identifier. The author in [7] projects a Rock Hyrax Swarm Optimizer with a DL-based Android malware detection (RHSODL-AMD) approach. Moreover, the Adamax enhancer with attention recurrent autoencoder (ARAE) technique has been used for Android malware recognition.

In [15], a pre-trained CNN technique in Android malware recognition is projected. An EfficientNet-B4 CNN-based method has been developed to recognize Android malware. These features are transferred by a worldwide average pooling layer and provided as a softmax identifier. The authors [16] presented a novel DL-based technique. Primary, it visualization a portable executable (PE) file as a colored image. Then, it identifies malware that relies on deep features employing support vector machine (SVM). The projected model integrates DL with ML methods and is observed by using benchmark datasets.

III. THE PROPOSED MODEL

In this paper, we have proposed an innovative IPR-EODL methodology for efficient and automatic Android Malware Detection. The purpose of the IPR-EODL technique is to properly identify and categorize the Android malware in such a way that security can be achieved. In the IPR-EODL technique, a three-stage process is involved namely data pre-processing, CA-LSTM-based Android malware recognition, and EO-based hyperparameter tuning. Fig. 1 portrays the complete workflow of the IPR-EODL system.

A. DATA PRE-PROCESSING

In the IPR-EODL technique, the data pre-processing step has been applied to transform input data into an attuned format. In classification, it can be essential to elect features for signifying the class the novel record would affect [13]. According to this, the permit and API calls have been eliminated from every Android app. Androguard [17] suggests a comprehensive package device intended to connect with Android files and controlled only by Python environments. The Androguard device has been deployed for investigating Android Package Kit (APK) files by separately eliminating the DEX file permit for every APK file. Therefore, it can create a data frame involving rows (apps) and columns (features), but every column suggests particular permits or API calls with dual values, and then rows refer the both benign and malware APK files.

B. DETECTION USING THE CA-LSTM MODEL

In this phase, the pre-processed data is passed into the CA-LSTM algorithm for the Android malware detection process. The CA-LSTM technique excels in capturing long-range dependencies and intricate patterns in the sequential data, making it very suitable for the dynamic and developing nature of Android malware. The combination of channel attention mechanisms improves the model's concentration on salient features, enabling it to recognize subtle variations that can represent malicious behavior. Moreover, the CA-LSTM approach's ability to automatically adjust to distinct levels of significance in various channels improves its robustness in detecting various features of Android malware. By leveraging the strengths of CA-LSTM, the IPR-EODL method drives to attain a sophisticated and effectual detection model able to handle the difficult and developing landscape of Android malware attacks.

LSTM network is developed as a new approach to address the problem of RNN [18]. This network constructs on RNN with the long-time delay process that could efficiently capture the relationship among longer series, helping to alleviate the gradient explosion or gradient disappearance. The LSTM exploits stored data to learn long-term dependency that is read, saved, and written via three gating processes like forget input, and output gates. The input gate is used to manage the data entering the network, the forget gate is used to regulate the retention of memory cells, and the output gate can be employed to regulate the output of the network. Among them, the forget gate is the most prominent; decides whether memory in the network will be maintained or removed which gives a long-term memory function. During the iteration procedure of a lithium-ion battery's charge or discharge cycle, there is often a capacity rebound process. But, the forget gate selectively forgets the memory, making the capacity rebound model improve the predictive performance and have a lesser effect on the training of the model.

The equation for the forget gate is given below:

$$f_t = \sigma\left(w_f \cdot \left[h_{t-1'} x_t\right] + b_f\right) \tag{1}$$

First, we splice the existing time step input x_t with prior time step output h_{t-1} to attain $[h_{t-1}, x_t]$ and later convert it through the fully connected (FC) layer. Lastly, by activating the sigmoid function, we attain the memory decay coefficient f_t. The sigmoid function is used to compress the output within

[0,1] and regulate the value that flows by the network.

$$i_t = \sigma\left(w_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{2}$$

$$\tilde{C} = \tanh\left(w_c \cdot [h_{t-1}, x_t] + b_c\right) \tag{3}$$

Afterward, a sequence of alterations from the FC layer attains the learned memory decay co-efficient , attained by stimulating the sigmoid function. In Eq. (3), it attains the input for the existing time-step x_t merged with prior time-step output h_{t-1} to attain $[h_{t-1}, x_t]$. Next, $[h_{t-1}, x_t]$ is transformed via the FC state. Finally, the present layer of learned memory $\tilde{C}$ is attained by the tanh function. The tanh activation function is used to compress the output within $[-1, 1]$, which limits the value flow by the network.

The equation for the cell layer upgrade is given below:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C} \tag{4}$$

By multiplying the decay co-efficient f_t through the memory state C_{t-1} of the prior time step, the C_t memory now is attained, besides the present memory decline co-efficien $_t$ by the presently learned memory $\tilde{C}$.

$$o_t = \sigma\left(w_0\left[h_{t-1'} x_t\right] + b_0\right) \tag{5}$$

$$h_t = o_t \cdot tanh\left(C_t\right) \tag{6}$$

The output gate co-efficient 0_t is attained by the sigmoid function. In Eq. (6), the output h_t of a single cell of the LSTM model now is attained.

In recent times, the attention mechanism (AM) has the considerable advances in DL. The attention-based method has become more popular in academia and industry for its effectiveness and interpretability. Initially, AM is based on the study of human vision. It is well developed that humans have a restricted capability for data processing in the field of cognitive science, thus selectively attending to any accessible data while avoiding the rest. Integrating AM with the technique makes it quickly detect higher value data from a great deal of data, decreases the effect of insignificant data, and optimizes the significance of data.

The study exploits the channel attention (CA) module squeeze and excitation (SE) block. The CA model could vigorously perfect the weight of features by absorbing the significance of feature networks and weighted the combination of features in dissimilar networks. Thus, the CA model enables to focus more on significant features and alleviates the effects of unimportant or redundant features. During the prediction of lithium battery RUL, the CA model could efficiently alleviate the effects of capacity and simultaneously enhance the use of features when data are restricted:

The CA model is split into three different stages namely squeeze operation, excitation operation, and scale operation.

At first, the squeeze function removes global spatial features from the network, which compress the spatial data into channel descriptors, and employing global average pooling, it creates statistics for all the channels. The equation for the squeeze operation is given in the following:

$$Z_c = F_{sq}\left(u_c\right) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c\left(i, j\right) \tag{7}$$

Next, the excitation operation completely captures the dependency for all the channels. The equation for the excitation operation is given below:

$$s = F_{ex}\left(z, W\right) = \sigma\left(g\left(z, W\right)\right) = \sigma\left(W_2 \delta\left(Wz\right)\right) \tag{8}$$

At last, the scale operation is used to multiply the weighted coefficients learned from all the channels with a novel feature to attain the weighted feature, which gives the model a detection capability for all the features. The equation for scale operation is given below:

$$\tilde{x}_c = F_{scale}\left(u_c, s_c\right) = s_c u_c \tag{9}$$

C. HYPERPARAMETER SELECTION USING THE EO ALGORITHM

Finally, the EO selects the hyperparameter values of the CA-LSTM approach. This amalgamation proposes to strike a balance among model complexity and performance by optimizing the CA-LSTM hyperparameter and making sure that the DL method efficiently captures intricate patterns and dependencies in Android malware data. The EO with its iterative optimizer procedure, adjusts the CA-LSTM method, permitting it to modify the unique features of the Android malware detection task.

The EO approach simulates the mass balance formula in a control volume in a physical standard and tries to determine the stability condition of systems [19]. EO is an effective optimization once resolving optimizer issues because of its upgrading process containing 3 stages such as initialization, equilibrium pool, and concentration upgrade.

Step 1: Initialization. During this stage, the position of all the particles can be assumed to the attention of control volume (C), but a group of particles has been created arbitrarily among the boundaries as:

$$C_{i,j} = c_{\min,j} + r\left(c_{\max,j} - c_{\min,j}\right), \quad i = 1,K,\ nj = 1,K,d \tag{10}$$

whereas $C_{i,j}$ signifies the position from the j^{th} size of *the* i^{th} particle, r denotes the random amount among (zero and one), $c_{\min,j}$, and $c_{\max j}$ denotes the boundaries of all the particles from j^{th} size. The fitness values (FVs) can be estimated and the particles can be arranged, the generation of initialization particles for constructing an equilibrium pool.

Step 2: Equilibrium candidate and pool. To determine the last equilibrium state, distinct best-far elements are needed to improve the population range. Hence, an equilibrium pool has been generated. Moreover, the average place of the above 4 particles has been computed and stored in the equilibrium pool. Next all the iterations, the 5 candidates were upgraded depending on the above process. The equilibrium pool was

expressed as:

$$C_{eq,pool} = \left\{C_{eq(1)}, C_{eq(2)}, C_{eq(3)}, C_{eq(4)}, C_{eq(ave)}\right\} \quad (11)$$

whereas $C_{eq,pool}$ denotes the equilibrium pool, $C_{eq(i)}(i = 1, 2, 3, 4)$ are 4 candidates with best so far FVs, and $C_{eq(ave)}$ implies the average position of 4 candidates that are formulated as:

$$C_{eq(ave)} = \frac{C_{eq(1)} + C_{eq(2)} + C_{eq(3)} + C_{eq(4)}}{4} \quad (12)$$

During all the iterations, the candidate has been chosen arbitrarily in the equilibrium pool as the optimum element from the existing iterations. It can be observed that all the candidates from the equilibrium pool have a similar probability to chosen, offering optimum diversity in population.

Step 3: Concentration upgrade. For updating the particle concentrations, the 2 major terms are assumed in the EO approach, generation rate (G) and exponential term (F). The term F challenge is to manage the stability among exploration as well as exploitation:

$$F = a_1 sign\,(r_1 - 0.5)\left[\exp\left(-r_2 t_{EO}\right) - 1\right] \quad (13)$$

In which, $r1$ and $r2$ represent the arbitrary vectors in the range of zero and one ; $a1$ denotes the constant for controlling the exploration capability, *and* t_{EO} implies the co-efficient of EO that can upgraded in all the iterations:

$$t_{EO} = (1 - T/M_{iter})^{(a_2 T/M_{iter})} \quad (14)$$

whereas, M_{iter} indicates the maximal iteration, T stands for the existing iteration, and $a2$ implies the constant adjusting of the exploitation capability. Once the $a1$ is superior, the exploration of the EO approach is enhanced. Also, if $a2$ is higher, the exploitation of EO is improved. During the primary EO approach, $a1$ and $a2$ denote the set as 1 and 2, correspondingly.

The rate of generation (G) is another essential term for upgrading in the EO system, it can be employed for transferring exact performance with improving exploitation. The mathematical process of the rate of generations (G) has been formulated as:

$$G = -P\left(C_{eq} - r_2 C_i\right) F, \quad i = 1, K, n \quad (15)$$

$$P = \begin{cases} 0.5 r_{d1} \cdot u & r_{d2} \geq GP \\ 0 & r_{d2} < GP \end{cases} \quad (16)$$

In which, C_{eq} denotes the elected candidate from the equilibrium pool; C_i stands for the position of i^{th} particle; $rd1$ and $rd2$ stand for the arbitrary numbers among zero and one; u implies the unit vector; and GP refers to the generation possibility that influences the exploitation and exploration are equivalent to 0.5.

So, the upgrading rule in EO is recognized as:

$$C_i^{new} = C_{eq} + \left(C_i - C_{eq}\right) F + \frac{(1 - F)\, G}{r_3 V} \quad (17)$$

whereas C_i^{new} denotes the upgrading position of *an* i^{th} particle, $r3$ defines the random vector from the array of zero, and one, V is equivalent to 1. Then the upgrading phase of EO checks the border of the upgraded locations and computes its FVs after deploying the memory-saving process for adopting optimum particles from the upgraded performances.

The EO algorithm develops a fitness function (FF) to undertake the optimum classifier result. It states an optimistic integer for implying the optimal solution of the candidate outcome. In this case, the reduction in classifier error rate has been regarded as FF, as written in Eq. (18).

$$\begin{aligned} fitness\,(x_i) &= Classifier\ Error\ Rate\,(x_i) \\ &= \frac{number\ of\ misclassified\ samples}{Total\ number\ of\ samples} * 100) \end{aligned} \quad (18)$$

IV. RESULTS AND DISCUSSION

In this part, the Android malware recognition outcome of the IPR-EODL methodology is verified using the database [20], [21], containing 7500 instances as defined in Table 1. A group of measures utilized for examining the classification outcomes is accuracy ($accu_y$), precision ($prec_n$), recall ($reca_l$), and F-score (F_{score}).

TABLE 1. Details on database.

Classes	No. of Instances
Benign	5000
Malware	2500
Total Instances	7500

$Prec_n$ evaluates the proportion of accurately predictive positive instances out of every instance that is forecasted as positive.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (19)$$

$Reca_l$ computes the proportion of positive instances appropriately classified.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (20)$$

$Accu_y$ measures the proportion of properly classified instances (positives and negatives) against the total instances (no. of instances classified).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (21)$$

F_{score} is an estimate integrating the harmonic mean of $prec_n$ and $reca_l$.

$$\text{F} - \text{score} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \quad (22)$$

Fig. 2 represents the confusion matrices generated by the IPR-EODL model below 80:20 and 70:30 of the training phase (TRAP)/testing phase (TESP). The results indicate the efficacious detection of the benign and malware samples below all classes.

In Table 2 and Fig. 3, the Android malware recognition result of the IPR-EODL technique is tested under 80:20 of

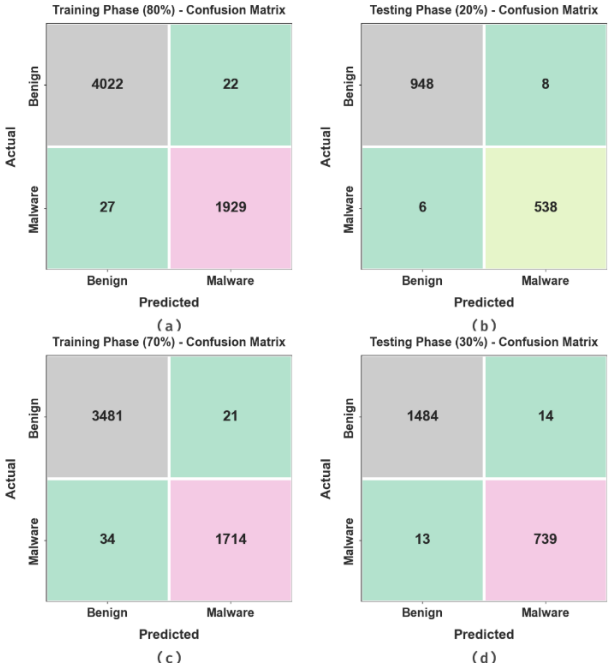

FIGURE 2. Confusion matrices of (a-c) TRAP of 80% and 70% and (c-d) TESP of 20% and 30%.

TABLE 2. Android malware recognition outcome of IPR-EODL methodology at 80:20 of TRAP/TESP.

Classes	$Accu_y$	$Prec_n$	$Reca_l$	F_{score}	MCC
TRAP (80%)					
Benign	99.18	99.33	99.46	99.39	98.14
Malware	99.18	98.87	98.62	98.75	98.14
Average	99.18	99.10	99.04	99.07	98.14
TESP (20%)					
Benign	99.07	99.37	99.16	99.27	97.98
Malware	99.07	98.53	98.90	98.72	97.98
Average	99.07	98.95	99.03	98.99	97.98

the TRAP/TESP. The experimental values emphasized that the IPR-EODL model appropriately categorizes benign and malware samples. Additionally, on 80% of the TRAP, the IPR-EODL method provides an average $accu_y$ of 99.18%, $prec_n$ of 99.10%, $reca_l$ of 99.04%, F_{score} of 99.07%, and MCC of 98.14%. Besides, with 20% of the TESP, the IPR-EODL technique delivers an average $accu_y$ of 99.07%, $prec_n$ of 98.95%, $reca_l$ of 99.03%, F_{score} of 98.99%, and MCC of 97.98% respectively.

In Table 3 and Fig. 4, the Android malware recognition study of the IPR-EODL method is definite at 70:30 of the TRAP/TESP. The outcome values stated that the IPR-EODL approach appropriately categorizes benign and malware samples.

Moreover, on 70% of the TRAP, the IPR-EODL method provides an average $accu_y$ of 98.95%, $prec_n$ of 98.91%, $reca_l$ of 98.73%, F_{score} of 98.82%, and MCC of 97.64%. Also, with

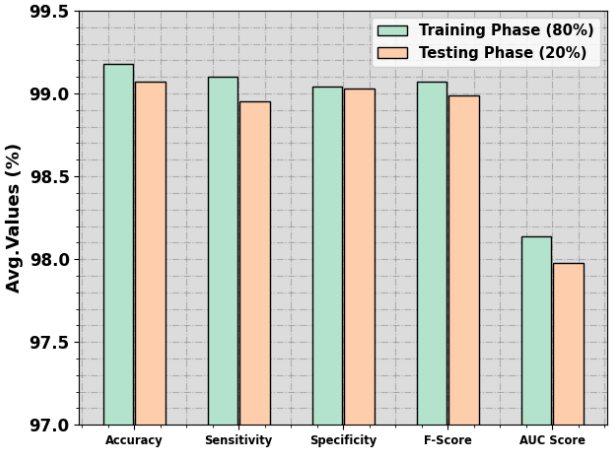

FIGURE 3. Average of IPR-EODL methodology at 80:20 of TRAP/TESP.

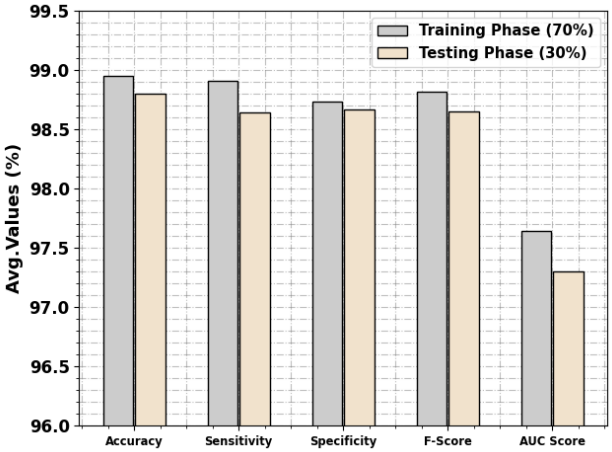

FIGURE 4. Average of IPR-EODL methodology at 70:30 of TRAP/TESP.

TABLE 3. Android malware recognition outcome of IPR-EODL methodology at 70:30 of TRAP/TESP.

Class	$Accu_y$	$Prec_n$	$Reca_l$	F_{score}	MCC
TRAP (70%)					
Benign	98.95	99.03	99.40	99.22	97.64
Malware	98.95	98.79	98.05	98.42	97.64
Average	98.95	98.91	98.73	98.82	97.64
TESP (30%)					
Benign	98.80	99.13	99.07	99.10	97.30
Malware	98.80	98.14	98.27	98.21	97.30
Average	98.80	98.64	98.67	98.65	97.30

30% of the TESP, the IPR-EODL model delivers an average $accu_y$ of 98.80%, $prec_n$ of 98.64%, $reca_l$ of 98.67%, F_{score} of 98.65%, and MCC of 97.30% correspondingly.

To estimate the performance of the IPR-EODL method under 80:20 of TRAP/TESP, TRA, and TES $accu_y$ curves are shown, as represented in Fig. 5. The TRA and TES

FIGURE 5. *Accu$_y$* curve of IPR-EODL methodology at 80:20 of TRAP/TESP.

FIGURE 6. Loss curve of IPR-EODL methodology at 80:20 of TRAP/TESP.

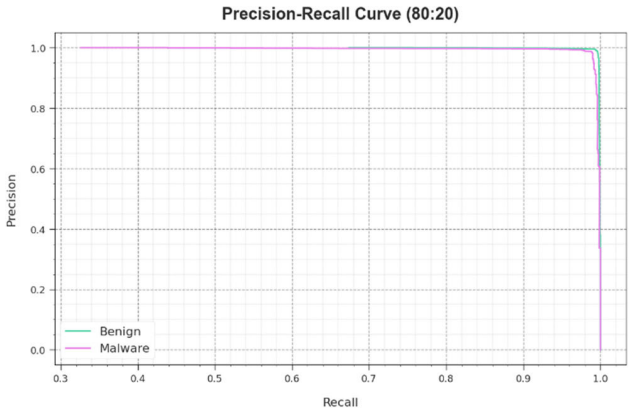

FIGURE 7. PR curve of IPR-EODL methodology at 80:20 of TRAP/TESP.

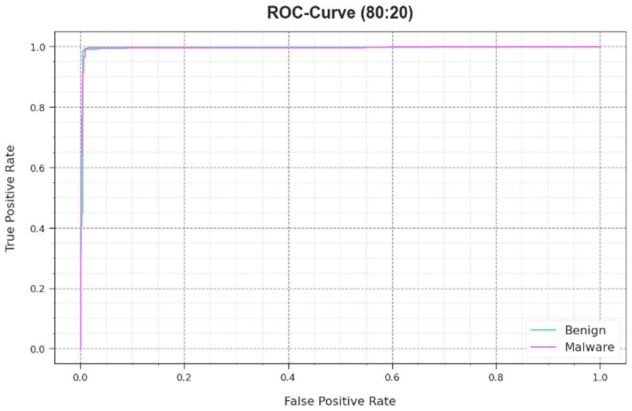

FIGURE 8. ROC curve of IPR-EODL methodology at 80:20 of TRAP/TESP.

accu$_y$ curves exhibit the solution of the IPR-EODL system over plentiful epochs. The figure delivers noteworthy details regarding the learning tasks and generalized capabilities of the IPR-EODL model. With a raised epoch amount, it is observed that the TRA and TES *accu$_y$* curves acquired increased. It is evidenced that the IPR-EODL methodology gets enhanced testing accuracy which can classify the designs in the TRA and TES data.

Fig. 6 illustrates the complete TRA and TES loss values of the IPR-EODL method at 80:20 of the TRAP/TESP over epochs. The TRA loss indicates the model loss is decreased over epochs. Primarily, the loss values are decreased as the model adjusts the weight to lessen the predictive error on the TRA and TES data. The loss curves reveal the amount to which the method fits the TRA data. It is perceived that the TRA and TES loss is gradually minimized and represented that the IPR-EODL model effectively learns the patterns demonstrated in the TRA and TES data. It is also clear that the IPR-EODL method adjusts the parameters to reduce the alteration among the forecast and original TRA labels.

The precision-recall (PR) curve of the IPR-EODL approach under 80:20 of TRAP/TESP is exhibited by plotting accuracy beside recall as determined in Fig. 7. The experimental values confirm that the IPR-EODL method gets improved precision-recall values on all 2 class labels. The figure describes that the model learns to recognize dissimilar class labels. The IPR-EODL methodology completes enriched results in the recognition of positive instances with fewer false positives.

The receiver operating characteristic (ROC) curves offered by the IPR-EODL model under 80:20 of TRAP/TESP are established in Fig. 8, which can separate the classes. The figure states valuable visions into the trade-off among the TPR as well as FPR rates over various classification thresholds and changing amounts of epochs. It offers the accurate forecast performance of the IPR-EODL technique on the identification of all 2 class labels.

In Table 4 and Fig. 9, an extensive comparison result of the IPR-EODL technique is provided [22]. The results indicate the supremacy of the IPR-EODL technique. Based on *accu$_y$* the IPR-EODL technique offers an increased *accu$_y$* of 99.18% while the J48, RF, decision tree (DT), Naıve Bayes (NB), Multilayer Perceptron (MLP), AdaBoostMI, and Automated Android Malware Detection using Optimal Ensemble Learning Approach for Cybersecurity (AAMD-OELAC) models obtain decreased *accu$_y$* of 96.80%, 97.80%, 94.60%, 69.10%, 98.10%, 88.40%, and 98.93% respectively.

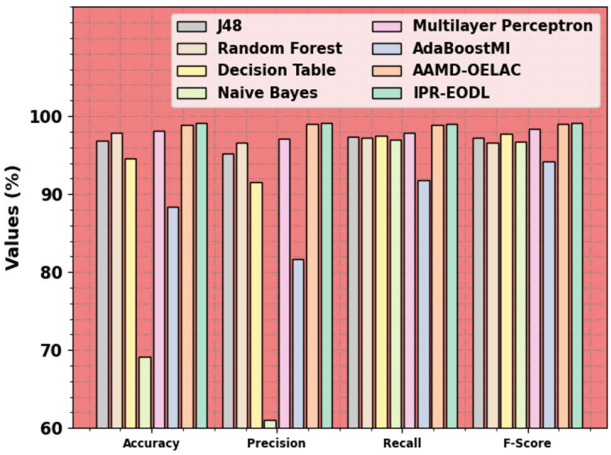

FIGURE 9. Comparative outcome of IPR-EODL methodology with other systems.

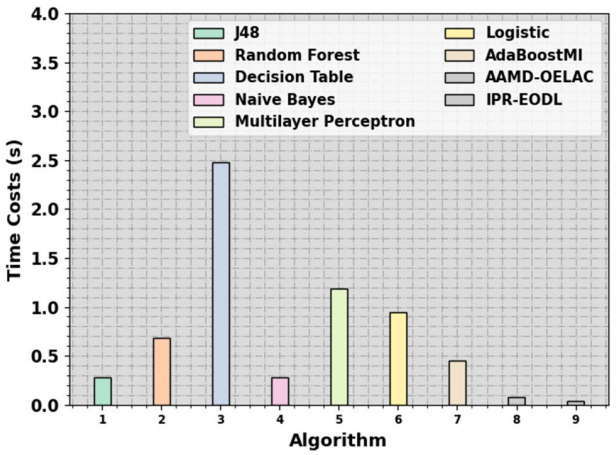

FIGURE 10. Comparative outcome of IPR-EODL methodology with other systems.

TABLE 4. Comparative outcome of IPR-EODL methodology with other systems [22].

Algorithm	$Accu_y$	$Prec_n$	$Reca_l$	F_{score}
J48	96.80	95.20	97.42	97.24
RF	97.80	96.60	97.22	96.62
DT	94.60	91.60	97.52	97.77
NB	69.10	61.00	96.92	96.71
MLP	98.10	97.10	97.90	98.35
AdaBoostMI	88.40	81.70	91.83	94.25
AAMD-OELAC	98.93	99.05	98.93	99.04
IPR-EODL	99.18	99.10	99.04	99.07

Meanwhile, based on $prec_n$ the IPR-EODL method gives raised $prec_n$ of 99.10% whereas the J48, RF, DT, NB, MLP, AdaBoostMI, and AAMD-OELAC techniques get minimized values $prec_n$ of 95.20%, 96.60%, 91.60%, 61.00%, 97.10%, 81.70%, and 99.05%. Eventually, based on $reca_l$ the IPR-EODL approach provides higher values $reca_l$ of 99.04% but the J48, RF, DT, NB, MLP, AdaBoostMI, and AAMD-OELAC models acquire reduced values $reca_l$ of 97.42%, 97.22%, 97.52%, 96.92%, 97.90%, 91.83%, and 98.93% separately.

TABLE 5. Comparative outcome of IPR-EODL methodology with other systems.

Algorithm	Time Costs (s)
J48	0.28
RF	0.68
DT	2.48
NB	0.28
MLP	1.19
Logistic	0.95
AdaBoostMI	0.45
AAMD-OELAC	0.08
IPR-EODL	0.04

In Table 5 and Fig. 10, the computation time (CT) cost analysis of the IPR-EODL technique with present models has been demonstrated. The results highlighted that the IPR-EODL method gains a lower CT of 0.04s. On the other hand, the J48, RF, DT, NB, MLP, AdaBoostMI, and AAMD-OELAC models accomplish increased CT values of 0.68s, 2.48s, 0.28s, 1.19s, 0.95s, 0.45s, and 0.08s respectively. Therefore, the IPR-EODL technique exhibits better performance than other models.

V. CONCLUSION

In this paper, we mainly concentrated on and developed an innovative IPR-EODL method for effectual and automatic Android Malware Recognition. The purpose of the IPR-EODL technique is to properly identify and categorize the Android malware in such a way that security can be achieved. In the IPR-EODL technique, a three-stage process is involved namely data preprocessing, CA-LSTM-based Android malware detection, and EO-based hyperparameter tuning. In this work, the IPR-EODL technique has exploited the CA-LSTM model for the recognition of Android malware. To enhance the performance of the CA-LSTM approach, the IPR-EODL technique employs the EO algorithm for the hyperparameter tuning method. The experimentation evaluation of the IPR-EODL technique was verified on a benchmark Android malware database. Extensive outcomes highlight the significant performance of the IPR-EODL technique on the Android malware detection process with a maximum accuracy of 99.18%. Future work can focus on the design of an ensemble classifier for an enhanced Android malware detection process.

ACKNOWLEDGMENT

The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University for funding this work through large group Research Project under grant number (RGP2/248/44). Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2024R237), Princess Nourah bint Abdulrahman

University, Riyadh, Saudi Arabia. Research Supporting Project number (RSPD2024R787), King Saud University, Riyadh, Saudi Arabia. This study is supported via funding from Prince Sattam bin Abdulaziz University project number (PSAU/2023/R/1444). This study is partially funded by the Future University in Egypt (FUE).

REFERENCES

[1] A. Gómez and A. Muñoz, "Deep learning-based attack detection and classification in Android devices," *Electronics*, vol. 12, no. 15, p. 3253, Jul. 2023.
[2] Y. Zhao, L. Li, H. Wang, H. Cai, T. F. Bissyandé, J. Klein, and J. Grundy, "On the impact of sample duplication in machine-learning-based Android malware detection," *ACM Trans. Softw. Eng. Methodol.*, vol. 30, no. 3, pp. 1–38, Jul. 2021.
[3] H. Wang, W. Zhang, and H. He, "You are what the permissions told me! Android malware detection based on hybrid tactics," *J. Inf. Secur. Appl.*, vol. 66, May 2022, Art. no. 103159.
[4] H. Rathore, A. Nandanwar, S. K. Sahay, and M. Sewak, "Adversarial superiority in Android malware detection: Lessons from reinforcement learning based evasion attacks and defenses," *Forensic Sci. Int., Digit. Invest.*, vol. 44, Mar. 2023, Art. no. 301511.
[5] V. Sihag, M. Vardhan, P. Singh, G. Choudhary, and S. Son, "De-LADY: Deep learning-based Android malware detection using Dynamic features," *J. Internet Serv. Inf. Secur.*, vol. 11, no. 2, pp. 34–45, 2021.
[6] M. Ibrahim, B. Issa, and M. B. Jasser, "A method for automatic Android malware detection based on static analysis and deep learning," *IEEE Access*, vol. 10, pp. 117334–117352, 2022.
[7] A. Albakri, F. Alhayan, N. Alturki, S. Ahamed, and S. Shamsudheen, "Metaheuristics with deep learning model for cybersecurity and Android malware detection and classification," *Appl. Sci.*, vol. 13, no. 4, p. 2172, Feb. 2023.
[8] A. Batouche and H. Jahankhani, "A comprehensive approach to Android malware detection using machine learning," in *Information Security Technologies for Controlling Pandemics*, 2021, pp. 171–212.
[9] P. Bhat and K. Dutta, "A multi-tiered feature selection model for Android malware detection based on feature discrimination and information gain," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 10, pp. 9464–9477, Nov. 2022.
[10] L. Hammood, I. A. Dogru, and K. Kiliç, "Machine learning-based adaptive genetic algorithm for Android malware detection in auto-driving vehicles," *Appl. Sci.*, vol. 13, no. 9, p. 5403, Apr. 2023.
[11] R. Raphael and P. Mathiyalagan, "Intelligent hyperparameter-tuned deep learning-based Android malware detection and classification model," *J. Circuits, Syst. Comput.*, vol. 32, no. 11, Jul. 2023, Art. no. 2350191.
[12] M. N. AlJarrah, Q. M. Yaseen, and A. M. Mustafa, "A context-aware Android malware detection approach using machine learning," *Information*, vol. 13, no. 12, p. 563, Nov. 2022.
[13] O. N. Elayan and A. M. Mustafa, "Android malware detection using deep learning," *Proc. Comput. Sci.*, vol. 184, pp. 847–852, Jan. 2021.
[14] P. Yadav, N. Menon, V. Ravi, S. Vishvanathan, and T. D. Pham, "A two-stage deep learning framework for image-based Android malware detection and variant classification," *Comput. Intell.*, vol. 38, no. 5, pp. 1748–1771, Oct. 2022.
[15] P. Yadav, N. Menon, V. Ravi, S. Vishvanathan, and T. D. Pham, "EfficientNet convolutional neural networks-based Android malware detection," *Comput. Secur.*, vol. 115, Apr. 2022, Art. no. 102622.
[16] K. Shaukat, S. Luo, and V. Varadharajan, "A novel deep learning-based approach for malware detection," *Eng. Appl. Artif. Intell.*, vol. 122, Jun. 2023, Art. no. 106030.
[17] A. Desnos and G. Gueguen, "Androguard documentation," Tech. Rep., 2018.
[18] C. Chen, J. Wei, and Z. Li, "Remaining useful life prediction for lithium-ion batteries based on a hybrid deep learning model," *Processes*, vol. 11, no. 8, p. 2333, Aug. 2023.
[19] C. Zhong, G. Li, Z. Meng, H. Li, and W. He, "A self-adaptive quantum equilibrium optimizer with artificial bee colony for feature selection," *Comput. Biol. Med.*, vol. 153, Feb. 2023, Art. no. 106520.
[20] *Andro-AutoPsy*. Accessed: Jul. 14, 2023. [Online]. Available: http://ocslab.hksecurity.net/andro-autopsy
[21] J.-W. Jang, H. Kang, J. Woo, A. Mohaisen, and H. K. Kim, "Andro-AutoPsy: Anti-malware system based on similarity matching of malware and malware creator-centric information," *Digit. Invest.*, vol. 14, pp. 17–35, Sep. 2015.
[22] H. Alamro, W. Mtouaa, S. Aljameel, A. S. Salama, M. A. Hamza, and A. Y. Othman, "Automated Android malware detection using optimal ensemble learning approach for cybersecurity," *IEEE Access*, vol. 11, pp. 72509–72517, 2023.