scientific reports

Check for updates

OPEN

Deep learning-based improved transformer model on android malware detection and classification in internet of vehicles

Naif Almakayeel

With the growing popularity of autonomous vehicles (AVs), confirming their safety has become a significant concern. Vehicle manufacturers have combined the Android operating system into AVs to improve consumer comfort. However, the diversity and weaknesses of the Android operating system pose substantial safety risks to AVs, as these factors can expose them to threats, namely Android malware. The advanced behaviour of multi-data source fusion in autonomous driving models has mitigated recognition accuracy and effectualness for Android malware. To efficiently counter new malware variants, novel techniques distinct from conventional methods must be utilized. Machine learning (ML) techniques cannot detect every new and complex malware variant. The deep learning (DL) model is an efficient tool for detecting various malware variants. This manuscript proposes a Deep Learning-Based Improved Transformer Model on Android Malware Detection (DLBITM-AMD) technique for Internet vehicles (IoVs). The main aim of the presented DLBITM-AMD approach is to detect Android malware effectually and accurately. The DLBITM-AMD method performs a Z-score normalization process to convert the raw data into a standard form. Then, the DLBITM-AMD approach utilizes the binary grey wolf optimization (BGWO) model to select optimum feature subsets. An improved transformer is integrated with the RNN model and softmax to enhance classification for Android malware recognition. Finally, the snake optimizer algorithm (SOA) method is employed to select the optimum parameter for the classification method. An extensive experiment of the DLBITM-AMD method is accomplished on a benchmark dataset. The performance validation of the DLBITM-AMD technique portrayed a superior accuracy value of 99.26% over existing Android malware recognition models.

Keywords Android malware detection, Internet of vehicles, Softmax, Snake optimizer algorithm, Deep learning

AVs have commonly increased in actual environments, acknowledged for increasing investigation and quick development on the IoVs framework[1]. The IoV can be essential to vehicular ad-hoc design, which permits communication using other vehicles through the Internet. External vehicle networks (EVNs) and Intra-vehicle networks (IVNs) establish most of the IoV. IoVs need various electronic control units (ECUs). However, as new vehicles become continuously more linked, their weaknesses in cyber-attacks are increasing[2]. Cyber-attacks affect the performance and stability of IoV and result in traffic mortalities or car outages. Various intruders betrayed a jeep automobile by performing risky actions, namely stirring the wheel steering and appealing the hand brake at a higher velocity, resulting in significant accidents. The malware computer operators and cyber-criminals often aim at self-driving vehicles or automobiles[3]. Everyone knows that linked vehicles function only in limited surroundings and can get only remote-control instructions by accepted secure communication channels like servers prepared by special software or manufacturers. A restricted atmosphere prevents the problem of unauthorized or malicious instructions[4]. Alternatively, new independent vehicles can connect to another vehicle, framework, and intellectual devices separate the vehicle by controlling internal information and signals. Motor vehicle safety is critical right now; hence, it is necessary to inspect all attack vectors upon vehicles with secure and remote cybersecurity[5]. The protection of self-driving vehicle holders is closely connected to the vehicle's safety. Earlier studies have established vehicle entries, which permit only official communication to the cars, and

Department of Industrial Engineering, College of Engineering, King Khalid University, 61421 Abha, Saudi Arabia. email: halmakaeel@kku.edu.sa

presented vehicle Intrusion Detection Systems (IDSs) for detecting strange behaviours in the Controller Area Network (CAN). Still, it is problematic in IDS or gateways to prevent this action, as most adware and malware are based on behaviour. To detect unidentified threats, it is essential to present a technique to identify unusual behaviours and examine atypical pointers with data analysis techniques[6].

Compared with traditional methods, namely signature-relevant malware recognition that depends on identifying specific paradigms of identified malware, ML-associated recognition can detect earlier undetectable malware categories and provide better recognition efficacy and efficiency. Some previous trainings have considered Android malware identification approaches associated with ML. However, specific limitations in the surveyed studies involved the present out-of-date works of literature based on prior analyses, the absence of conversation, and the narrow range of training regarding few debated content[7]. Deeply knowing the values, tasks and security, architecture detection, and upcoming study development tendencies of Android malware detection using DL, and understanding the novel trends of foreign and domestic investigation, it is essential to utilize Android malware recognition associated DL techniques[8]. As AVs become increasingly popular, the requirement for robust safety measures in the IoVs framework has increased substantially. Incorporating advanced communication networks in AVs has heightened their exposure to cyber threats, requiring efficient malware detection and classification. Addressing these safety threats is significant for ensuring the safety and reliability of AV systems in the swiftly growing IoV landscape[9]. The rise of connected vehicles and their integration into the broader Internet of Vehicles (IoV) network has heightened the risk of cyber threats and malware attacks. As these systems become increasingly advanced, they present new opportunities for malicious actors to exploit vulnerabilities, potentially jeopardizing vehicle safety and operational integrity. Addressing these threats needs advanced detection and classification methodologies that can efficiently detect and reduce malware risks. By developing enhanced deep learning models specifically custom-made for the intrinsic and dynamic environment of IoVs, the security and reliability of autonomous vehicles can be improved, confirming safer transportation systems[10].

Contribution of the proposed model

This manuscript proposes a Deep Learning-Based Improved Transformer Model on Android Malware Detection (DLBITM-AMD) technique for Internet vehicles (IoVs). The main aim of the presented DLBITM-AMD approach is to detect Android malware effectually and accurately. The DLBITM-AMD method performs a Z-score normalization process to convert the raw data into a standard form. Then, the DLBITM-AMD approach utilizes the binary grey wolf optimization (BGWO) model to select optimum feature subsets. An improved transformer is integrated with the RNN model and softmax to enhance classification for Android malware recognition. Finally, the snake optimizer algorithm (SOA) is employed to select the optimum parameter for the classification method. An extensive experiment of the DLBITM-AMD method is accomplished on a benchmark dataset. The major contribution of the DLBITM-AMD method is listed below.

- The DLBITM-AMD technique employs Z-score normalization to standardize feature values, confirming that all features contribute equally by removing biases from differing scales. This model improves the method's performance and accuracy by providing a consistent scale for analysis and enhancing the efficiency of subsequent algorithms.
- The DLBITM-AMD model utilizes BGWO-based feature selection (FS) to identify and choose the most relevant features from the dataset. This process substantially improves the model's performance by reducing dimensionality and focusing on the most appropriate attributes, paving the way to more effectual and accurate evaluation.
- An enhanced transformer-RNN-based classification incorporates advanced transformers with RNNs to attain greater classification accuracy and robust sequence modelling. This hybrid methodology implements the merits of both architectures, resulting in more precise and effective handling of intricate patterns and dependencies in data.
- The DLBITM-AMD methodology implements SOA-based hyperparameter tuning to optimize the model's performance by finding the most efficient parameter settings. This technique improves the accuracy and efficiency of the classification by systematically exploring and fine-tuning hyperparameters to attain optimal outcomes.
- The incorporated optimization model combines advanced models such as Z-score normalization, BGWO-based FS, enhanced transformer-RNN classification, and SOA for hyperparameter tuning—into a unified framework. This novel technique presents an overall solution for data preprocessing, FS, classification, and model optimization, attaining superior performance and efficiency across overall analysis phases.

Related works

In[11], a DL method was developed to incorporate the convolutional neural network (CNN) characteristics and effectually seize the malicious software's complex features. Furthermore, optimizer methods are utilized to enhance the presentation of the method, and a CNN DL method based on learning rate adjusting Adam is present. Adjusting the momentum decay and learning rates. The RNNs model the mined image features sequence to seize temporal dependency among malicious software. In[12], a novel multimodal DL Android malware detection technique called Chimera is presented, which integrates both the automatic and manual feature engineering by utilizing the DL structures deep neural network (DNN), Transformer Networks (TNs), and CNN methods to implement feature learning correspondingly. The technique also implements the Knowledge Discovery in Databases (KDD) method and utilizes the freely accessible Android benchmark datasets. Almutlaq et al.[13] developed a dual-phase IDS in ITS. For instance, the presented IDS model utilizes the extraction rule methods from DL methods. DNN in 2 phases. In the initial phase, the method examines the network traffic. The next

phase is raised to recognize the attack type. The method develops 3 rule extraction variants. The initial and the following variations are homogeneous, and they use the HypInv rule and DeepRed mining techniques in both phases individually. The third variation is heterogeneous, and it uses HypInv in the initial phase to execute binary classifications and DeepRed in the second phase to execute attack classifications. In[14], VoteDroid, a new fine-tuned DL method-based ensemble voting classifier, was presented. Also, this method presented using the random search optimizer methods for determining the framework of the method is utilized as voter classifiers in the ensemble classifiers. This optimizer technique is used to create 3 various DL methods: pure ANN, CNN-ANN, and pure CNN. Then, the method recommended hybridizing the three fine-tuned DL methods to make one ensemble voting classifier with two various work methods, such as Label Majority Rule (LMR) and Malware Minority Rule (MMR). Luo et al.[15] present a multilayer IDS structure containing AI-based and rule-based components. The rule-based component is utilized to identify the communication method, message interval SOME/IP-SD message, and SOME/IP header. The AI-based module acted on the payloads. Also, the method presents a SOME/IP dataset creation technique to compute the presented multilayer IDS performances.

Al-Hawawreh and Moustafa[16] introduce an attack intelligence framework for recognizing cyber-physical attacks and mining attack intelligence. Also, the method presents an attribution module for attack recognition by utilizing several ML and DL methods. The technique also uses Explainable AI (XAI) to enhance the explainability of the attack attribution component and mine attack intelligence. Abdullah et al.[17] emphasize a hybrid static classification based on CNN and bidirectional LSTM methods for Malware classifier tasks in the IoT. This method takes and learns the nature of annotation, Byte complex patterns, and assembly files, which are represented in single-dimensional images to provide superior feature extractions. The CNN method is utilized for automated feature extraction and selection. Additionally, the mined features are delivered to the bidirectional LSTM for classifications. Pravin et al.[18] presented a new Deep Q-learning network-based circle search (DQL-CS) method. First, the dataset is preprocessed using the Term Frequency Inverse Document Frequency (TF-IDF) methods, and the gathered feature sequence is mined using n-gram sampling. This method forms the classifiers to improve the classification and detection performances. Afterwards, extraction processes and the classification method are performed using the presented DQL-CS method. Alguliyev, Aliguliyev, and Sukhostat[19] integrate visual representations and transfer learning (TL) techniques to enhance malware detection. It employs pre-trained networks such as AlexNet and the MobileNet approach for classifying malware from grayscale images, with Radon transform employed to improve the accuracy. In[20], a Vision Transformer (ViT)-based malware detection technique is proposed, which utilizes attention maps for high accuracy and interpretability. The model processes application images to produce attention maps, emphasizing key features and extracting class and method names depending on these factors.

Ahmad et al.[21] propose a classification technique by utilizing a support vector machine (SVM) model, optimized with three approaches: Nuclear Reactor Optimization (NRO), Artificial Rabbits Optimization (ARO), and Particle Swarm Optimization (PSO). Zhan et al.[22] introduce an adversarial robustness anomaly detection technique that evaluates behaviour units to improve robustness. The method reduces perturbation attacks on low-level and high-level behaviour logs by extracting and analyzing behaviour units with semantic data and utilizing a multilevel DL method to comprehend behaviour semantics and context. In[23], an IoT device classification and attack detection system are introduced using SDN-enabled FiWi IoT networks. It features dynamic bandwidth allocation with a hybrid neural network, preprocessing traffic data, and feature extraction. Optimal features are chosen by utilizing chaotic seagull optimization (CSO). IoT/non-IoT classification employs a transformer model, and attack detection is handled by a novel slice attention-based deep capsule autoencoder (SA_DCAE) method. Wang et al.[24] propose an intellectual digital twin methodology for IoT attack behaviour detection, employing spatiotemporal feature fusion. It features information gain-based dimensionality reduction, a parallel spatiotemporal extraction model integrating CNN, bidirectional long short-term memory (BiLSTM), attention mechanisms, and DNN models. Egitmen et al.[25] present an opcode-based technique that employs several behavioural target variables to improve static malware classification. Zhao et al.[26] propose a masked autoencoder (MAE)-based traffic transformer with multilevel flow representation. The model employs a hierarchical traffic matrix and features packet-level and flow-level attention for effectual feature extraction. The MAE methodology pre-trains the classifier on unlabeled data and fine-tunes it with labelled data for traffic classification.

Mai et al.[27] propose a model Coll-DSVDD that integrates multi-agent collaboration with a deep support vector data description (DSVDD) model to optimize vehicular network security. Utilizing the deep convolutional networks and DSVDD in each network and enabling inter-network collaboration enhances defense against new attacks and lowers false negative rates without needing retraining. Sun et al.[28] introduce the CNN-LSTM with Attention Model (CLAM) technique for detecting intrusions in CAN networks. CLAM utilizes 1D convolution to extract features, the Bidirectional LSTM method to capture temporal dependencies, and attention mechanisms to accentuate key time steps, enhancing accuracy and convergence. It utilizes bit flip rates to delineate signal boundaries, removing the CAN communication matrix parsing requirement, and is compatible with different vehicles. In[29], the authors present the CCID-CAN model for CAN bus intrusion detection in autonomous vehicles. It utilizes a rule-based Valid Bit Index (VBIN) method for initial detection, followed by the Kalman filter and Naïve Bayes techniques to detect missed attacks. A cross-chain mechanism optimizes the Naïve Bayes detector by exchanging attack logs among connected AVs. Kishore and Behera[30] are to develop and evaluate a Deep Learning-based Convolutional Neural Network (CNN) technique for detecting attacks in Cyber-Physical Systems, specifically within connected autonomous vehicles. Maray et al.[31] introduce the IPR-EODL approach, which incorporates Equilibrium Optimization with Deep Learning for Android malware detection, employing channel attention LSTM for recognition and EO for hyperparameter tuning to enhance security and classification accuracy. Haouas et al.[32] propose a deep learning model with a six-layer architecture for detecting malware attacks in the network layer of medical IoT systems, constructed to operate within 2 ms and be deployable on devices with limited memory, thereby improving IoT network security and patient data protection.

The existing studies have various limitations. CNNs may slip certain malicious features due to their complexity. Techniques integrating automatic and manual feature engineering can suffer high computational costs and complexity. Dual-phase IDS systems may also encounter scalability and adaptability problems, while ensemble methodologies employing random search optimization may be more effective. Multilayer IDS structures may need help with combining AI-based and rule-based components. Frameworks utilizing diverse ML and DL models may face enhanced computational overhead. Static classification techniques could fail to address dynamic threats efficiently, and Deep Q-learning-based techniques might be computationally intensive. Moreover, techniques that depend on visual representations and TL may encounter threats with high computational demands and several kinds of malware. Present techniques often need help with scalability, real-time adaptability, and handling growing or various malware behaviours. There is a requirement for more effectual, adaptable, and comprehensive techniques that balance complexity with practical performance in dynamic environments.

The proposed method

This manuscript proposes the DLBITM-AMD technique for IoVs. The main aim of the DLBITM-AMD approach is to efficiently and accurately automate the recognition of Android malware. The DLBITM-AMD method comprises Z-score normalization, BGWO-based FS, Enhanced transformer-RNN-based classification, and an SOA-based hyperparameter tuning process to achieve this. Figure 1 demonstrates the workflow of the DLBITM-AMD technique.

Data normalization

Primarily, the DLBITM-AMD model implements a Z-score normalization procedure to measure the raw data into a standard form. Z-score normalization is a significant model for improving Android malware recognition in the IoV[33]. By altering data based on the standard and mean deviation, Z-score normalization certifies that features are even, enabling superior performance of ML techniques. This technique is mainly effective in the IoV environment, where data arises from numerous devices and sensors with dissimilar measures. Standardizing the data aids in precisely identifying malicious and anomaly patterns indicative of malware. Therefore, Z-score normalization decreases false positives and recognition accuracy and boosts the complete security of the IoV system, ensuring vehicles and their communication networks are protected from cyberattacks.

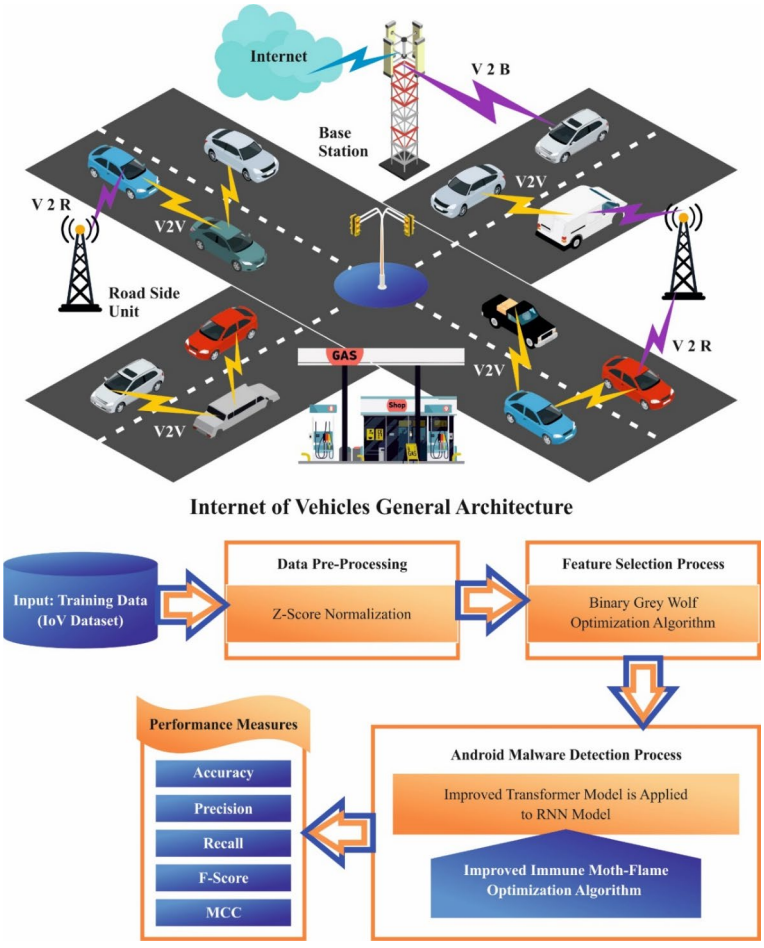

Fig. 1. Overall flow of DLBITM-AMD technique.

BGWO-based FS

Next, the DLBITM-AMD approach proposes a BGWO model to select optimum feature subsets[34]. The DLBITM-AMD approach utilizes the BGWO model to choose optimal feature subsets due to its efficiency in handling feature selection in high-dimensional spaces. The BGWO methodology, inspired by the hunting behaviour of grey wolves, effectually explores the feature space to detect the most relevant subsets by balancing exploration and exploitation. This method outperforms conventional technqiues by mitigating computational complexity and enhancing the performance of the model through more accurate feature subset detection. The input comprises of a set of features and their respective importance scores, while the output is the optimal subset of features that improves the classification model's accuracy and effectualness. For the GWO approach, wolves animatedly change their positions to detect and catch the prey efficiently. However, a few challenges, like FS, offer a binary space problem, but the performance has been controlled to values of both 0 and 1. This poses a challenge for the standard GWO approach. Figure 2 illustrates the workflow of the BGWO model.

Then, the BGWO approach is called conducting FS in tasks that contain solutions projected in the binary method.

$$x_d^{(t+1)} = \begin{cases} x_1^d, & if\ rand < \frac{1}{3} \\ x_2^d, & \frac{1}{3} \leq\ rand < \frac{2}{3} \\ x_3^d, & otherwise \end{cases} \tag{1}$$

Here, the update rule for $x_d^{(t+1)}$ is defined as follows: $x_d^{(t+1)}$ takes the value of x_1^d if a random value is less than 1/3, x_2^dd if the random value is between 1/3 and 2/3, and x_3^d, otherwise.

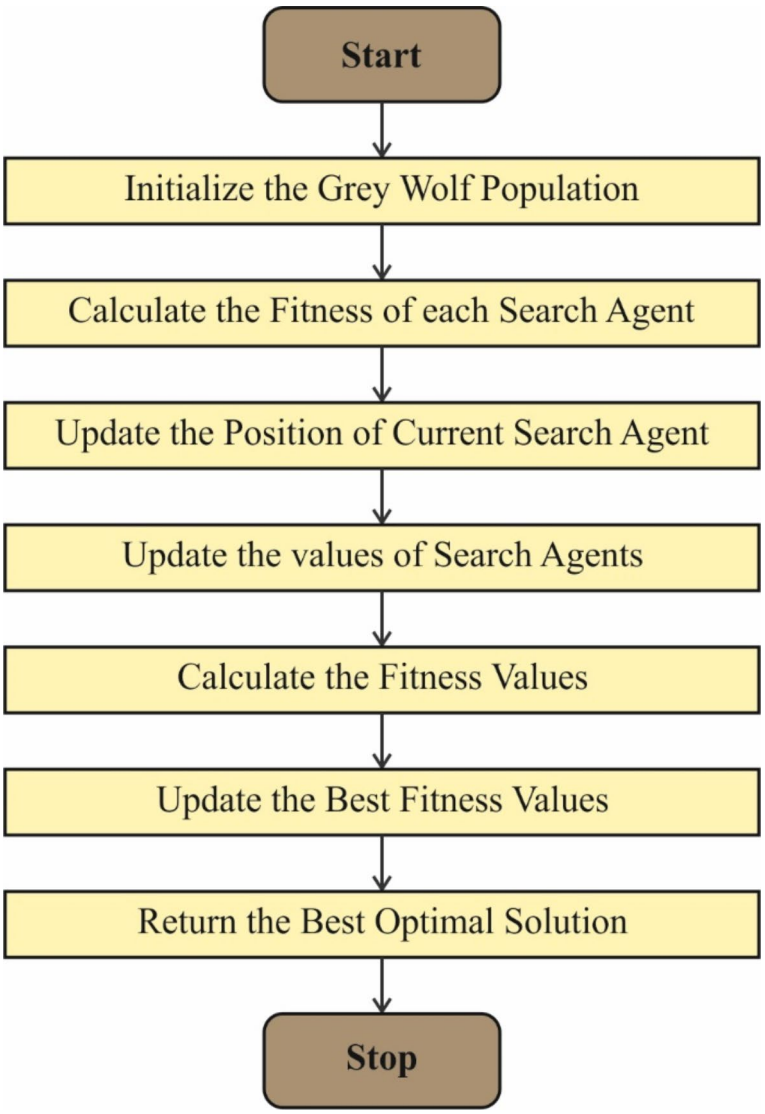

Fig. 2. Workflow of BGWO model.

$$x_d^{(r+1)} = \begin{cases} 1, & if\ sigmoid\ \left(\frac{x_1+x_2+x_3}{3}\right) \geq\ rand \\ 0, & otherwise \end{cases} \tag{2}$$

In this case, $rand$ signifies the random number from zero to one, which adapts to a uniform distribution. The variable x_d^{r+1} is the upgraded place of d-dimension dual wolf t^{th} iteration. The update rule for $x_d^{(r+1)}$ is as follows: $x_d^{(r+1)}$ is set to 1 if the sigmoid function applied to the average of x_1, x_2, and x_3 is greater than or equal to a random value and is set to 0 otherwise. The sigmoid is generally expressed as:

$$sigmoid\,(x) = \frac{1}{1+e^{-10(x-0.5)}} \tag{3}$$

x_1, x_2, and x_3 are dual vectors, which correspondingly represent the resultant of wolf action from the route of α, β, and δ grey wolves.

$$x_1^d = \begin{cases} 1, & if\ \left(x_\alpha^d + bstep_\alpha^d\right) \geq\ 1 \\ 0, & otherwise \end{cases} \tag{4}$$

$$x_2^d = \begin{cases} 1, & if\ \left(x_\beta^d + bstep_\beta^d\right) \geq\ 1 \\ 0, & otherwise \end{cases} \tag{5}$$

$$x_3^d = \begin{cases} 1, & if\ \left(x_\delta^d + bstep_\delta^d\right) \geq\ 1 \\ 0, & otherwsise \end{cases} \tag{6}$$

The α, β, and δ wolf's locations are represented as x_α^d, x_β^d, and x_δ^d, correspondingly. Furthermore, the values p_α^d, $bstep_\delta^d$, and $bstep_\beta^d$ are defined by

$$bstep_\alpha^d = \begin{cases} 1, & if cstep_\alpha^d \geq\ rand \\ 0, & otherwsise \end{cases} \tag{7}$$

$$bstep_\beta^d = \begin{cases} 1, & if cstep_\beta^d \geq\ rand \\ 0, & otherwsise \end{cases} \tag{8}$$

$$bstep_\delta^d = \begin{cases} 1, & if\ cstep_\delta^d \geq\ rand \\ 0, & otherwsise \end{cases} \tag{9}$$

The variables $cstep_\alpha^d$, $cstep_\beta^d$, and $cstep_\delta^d$ are defined as:

$$cstep_\alpha^d = \frac{1}{1+e^{-10\left(A_1^d D_\alpha^d - 0.5\right)}} \tag{10}$$

$$cstep_\beta^d = \frac{1}{1+e^{-10\left(A_1^d D_\beta^d - 0.5\right)}} \tag{11}$$

$$cstep_\delta^d = \frac{1}{1+e^{-10\left(A_1^d D_\delta^d - 0.5\right)}} \tag{12}$$

The values of A_1^d, D_α^d, D_β^d, and $D_\delta^d \delta$ are evaluated utilizing Eqs. (4), (5), (6).

Also, this BGWO has upgraded the data from any location. The solution is referred to as a 1D vector. Its size equals the feature counts. As part of this binary vector, the rates are zero and one.

The solution in the search is signified as a vector with 1D, but the size supports the feature counts. In the structure of this binary vector, the mathematical values 0 and 1 relate to the subsequent meanings:

0: The feature could not be elected.

1: The feature could be elected.

The FS procedure fundamentally proceeds with a dual-objective feature. One primary purpose is to reduce the feature counts, and the other is to enhance classifier precision. To attain both objectives concurrently, the FF integrates the following formulas.

$$fitness = \propto\ \rho_R(D) + \beta\,\frac{|S|}{|T|} \tag{13}$$

The parameters $\propto$ and β are determined as $\alpha == [0,1]$ and $\beta = (1-\alpha)$, respectively. The term $\rho_R(D)$ describes the error value. Additionally, $|S|$ implies the designated features subset, in which $|T|$ stands for all the features comprised in the data.

Improved transformer model

For Android malware recognition, an improved transformer is used for the RNN method, which is united with softmax for classification and offers various advantages. The transformer improves the capability of the RNN

technique for capturing long-range dependencies and complex patterns within malware data, which is significant in detecting complex threats. The softmax function gives a robust classification mechanism, translating model outputs into probability dispersions that facilitate accurate and interpretable outcomes. This incorporation enhances the accuracy and effectualness of the detection by employing the merits of both transformers and RNNs models, safeguarding a more efficient technique to detect and classify Android malware. The Transformer was projected and first used in natural language processing (NLP), where it attained extraordinary outcomes. The main structure of the Transformer includes a decoder and an encoder that varies from the recurrent structure in conventional recurrent neural networks (RNNs). Instead, it uses the self-attention mechanism, which allows equivalent computing and increases the computation efficiency of the method. Figure 3 depicts the infrastructure of the improved transformer.

RNN creates a novel network topology by analyzing old data and relating the hidden layer (HL) of dissimilar time eras in a time sequence together[35]. In this method, the RNN technique can discover a similar data construction, thus refining the prediction outcome by analyzing the past data. At the same time, the classical artificial neural network (ANN) may not contact the past data. The result of the RNN methodology comprises a mixture of past and state data of the HL, where the past data will impact the prediction outcome of the complete neural networks.

$$y_t = g\left(w_y h_t\right) \tag{14}$$

$$h_t = f\left(W_{x_t} + U h_{t-1}\right) \tag{15}$$

Here, y_t denotes the output value under time t; g and f denote the excitation functions in the RNN neural network; h_t represents the state of the HL under time t; W and U refer to the weight of the HL; w_y refers to the weight of the output layer.

The interior structure defines the effect of prediction, where a Tanh-style activation function might consist of the past HL data to the subsequent recurrent unit. Conversely, further parameters will be required to analyze the past data. Also, the RNN has a few disadvantages, such as inserting further parameters to attain superior past data analysis. It upsurges the complete calculation of the prediction method, which leads to extensive training time. Furthermore, the RNN network method seeks to enhance the parameter of training depending upon the calculated gradient value, which is challenging to define. So, it causes the gradient problem, which significantly affects the prediction accuracy of the method.

After obtaining the original input X_n, the Transformer creates a novel input X'_n with location encoding data over input embedding and location embedding processes[36]. If X'_n is gained, the self-attention block executes linear transformation to get Q (Query), K(key), and V (Value). Then, it computes the self-attention block output utilizing the below-mentioned calculations:

$$Attention\ (Q,\ K,\ V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{16}$$

$$MultiHead(Q,\ K,\ V) = Concat\ (hcad_1,\ head_2, \ldots\ ,\ head_n)\,W,$$

where

$$head_i = Attention\left(QW_i^Q,\ KW_i^K,\ VW_i^V\right) \tag{17}$$

Here, the self-attention mechanism can be classified into 3 phases. At first, the weight coefficient matrix is computed depending on the resemblance between Q and K. It is generally completed by calculating the dot product of matrix vectors, i.e., $F(Q,\ K) = QK^T$, or by computing the cosine resemblance among vectors. Next, the novel distribution of weights gained in the preceding step is regularized utilizing the $softmax$ function, which classifies every element into a likelihood distribution with some weights of 1. To alleviate the value and

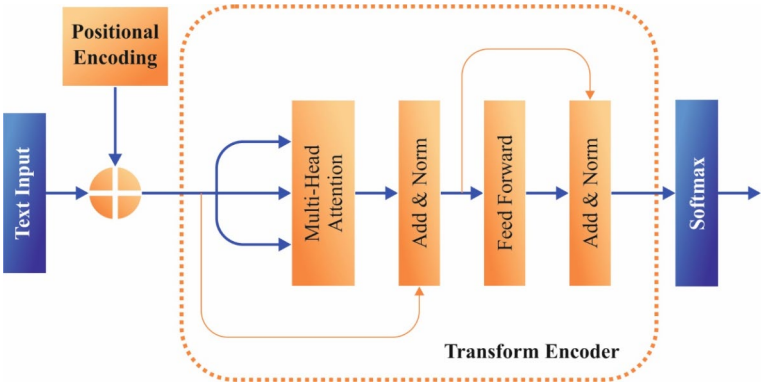

Fig. 3. Structure of improved transformer.

the gradient problem disappearance in $softmax$, the normalization factor $\frac{1}{\sqrt{d_k}}$ has been employed. Lastly, the values of weighted sum V are calculated depending upon the handled weight co-efficient matrix, resulting in the value of attention for Q.

The main feature of the Transformer is its multi-head attention mechanism, which is demonstrated in Eq. (4). The multi-head attention draws dissimilar attributes into manifold subspaces, permitting the method to progress data equally from assorted subspaces and enhancing the attention mechanism's performance. This method allows the Transformer to seize compound relationships amongst dissimilar input portions and efficiently model dependences over the sequence.

SOA-based parameter tuning

Finally, the SOA technique is developed to select the optimum parameter for the classification method[37]. The SOA approach is chosen for optimizing classification parameters due to its efficient exploration and adaptability in high-dimensional spaces. Unlike conventional techniques, the SOA model replicates snake behaviour to dynamically search for optimal solutions, averting local minima and effectually navigating intrinsic parameter landscapes. This adaptive strategy improves the accuracy of the classification methodology by giving robust and precise parameter selection. The input comprises initial parameter ranges and classification metrics, while the output consists of the optimal parameters that maximize classification performance, namely accuracy and precision. The SOA is a novel heuristic, intellectual optimizer technique that represents the snake's reproduction and mating. If the snake has enough food and lower temperature conditions, it attempts to discover the finest mate. Figure 4 portrays the flowchart of SOA.

Initialization

In SOA, the snake location relates to the optimizer issue solution; the individual area of the snake is set at random, and the modest function values signify the grade of excellence solution. Similar to heuristic techniques, the SOA produces an arbitrary population, and the formulation is given below:

$$X_i = X_{\min} + r \times \ (X_{\max} - X_{\min}) \tag{18}$$

Whereas x_i represents the snake's location, ris refers to a produced number at random, and X_{max} and $X_{\min}$ represent the upper and lower bounds, respectively.

Grouping of snakes

The population of snake N has been separated into dual sets: female and male. The mathematical formulation for both sets is given below:

$$N_m = \frac{N}{2} \tag{19}$$

$$N_f = N - N_m \tag{20}$$

whereas N denotes the total number of distinct snakes, N_m and N_f refer to the number of male and female snakes, respectively.

Discover the finest set of distinct in every cluster as per $temp$ and quantity of food; the $temp$ calculation is given below:

$$Temp = \exp\left(\frac{-l}{T}\right) \tag{21}$$

Here, l and T denote the current and highest iteration, respectively.

The quantity of food Q calculation has been expressed below:

$$Q = c_1 \times \ \exp\left(\frac{l-T}{T}\right) \tag{22}$$

While c_1 refers to a constant.

Exploration stage (no food)

When Q < TH1 (TH10.25), the snake hunts for food by picking places randomly and upgrading their position:

$$X_i\,(t+1) = X_r\,(t) \pm \ c_2 \times \ A \times \ ((X_{\max} - X_{\min}) \times \ r + X_{\min}) \tag{23}$$

$X_r\,(t)$ is the value of X_r at the current iteration, c_2 is a scaling constant, A is a factor affecting the adjustment magnitude, $X_{\max}$ and $X_{\min}$ are the bounds for X, and r is an arbitrary value between 0 and 1. This formula adjusts X_i based on $X_r\,(t)$, with the adjustment magnitude determined by a random value scaled by the range $[\,X_{\max} - X_{\min}]$.

In the Hunger Game Search (HGS) method, the operator $\pm \ is$ employed for foraging behaviour and distinct cooperative communication. The technique can achieve noble exploration in every probable route by delivering the prospect of decreasing or increasing the position solution.

A denotes the snake's capability to discover food and is formulated below:

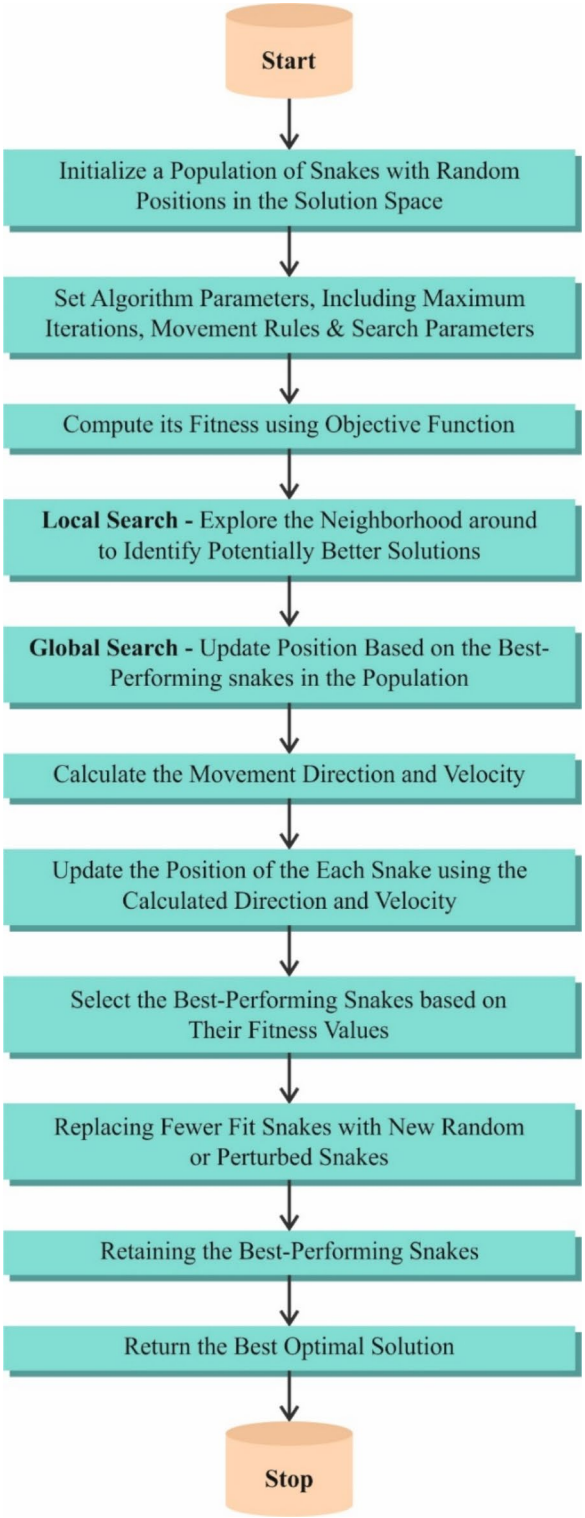

Fig. 4. Flowchart of SOA.

$$A = \exp\left(\frac{-f_r}{f_i}\right) \tag{24}$$

Here, f_r and f_i represent the X fitness value of selected snake locations at random; cis is a constant.

Exploitation stage (food exists)
In QTH1, the snake is hot when $Temp$> TH2(0.6). The position upgrade formulation is given below:

$$X_i(t+1) = X_{food} \pm c_3 \times Temp \times r \times (X_{food} - X_i) \tag{25}$$

X_{food} denotes the finest snake location, $c\,is$ a constant, $Temp$ denotes the scaling factor, r is a random factor, and X_i is the current updated value.

Fight and mating modes

The snake is in a chill state when $Temp <$ TH2. Also, it is both fight and mating modes.

The upgrade formulation is given below:

$$X_i(t+1) = X_i(t) + c_3 \times F \times r \times (Q \times X_{best,d} - X_i) \tag{26}$$

whereas $X_{best,d}$ refers to the best location amongst hetero-sexual snakes, F denotes the snake's fight skill, c_3 is a coefficient, r is a random factor, Q is the scaling factor applied to the best-known value $X_{best,d}$, $X_{best,d}$ is the best-known value for the dimension d, X_i is the current value being updated.

$$F = \exp\left(\frac{-f_{best,d}}{f_i}\right) \tag{27}$$

While $f_{best,d}$ refers to the fitness value of the finest snake's location $X_{best,d}$ amongst hetero-sexual snakes.

The mode of mating location upgraded calculation was given below:

$$X_i(t+1) = X_i(t) + c_3 \times M \times r \times (Q \times X_{best,d} - X_i) \tag{28}$$

where $X_i(t)$ is the value of X_i at the current iteration, c_3 is a scaling constant, M refers to the mating skill of a snake, r is a random value between 0 and 1, Q is a coefficient, and $X_{best,d}$ represents the best-known solution in the current dimension. This rule updates X_i by moving it towards $\times X_{best,d}$, with the movement scaled by c_3 , M and a random factor r.

$$M = \exp\left(\frac{-f_i}{f_{id}}\right) \tag{29}$$

Here, $f_{i,d}$ refers to the fitness value of ith hetero-sexual location. When the eggs are produced and picked, the worst females and males will be substituted:

$$X_{worst} = X_{\min} + r \times (X_{\max} - X_{\min}) \tag{30}$$

X_{worst} is the worst location in the snake cluster. Algorithm 1 illustrates the algorithm of the SOA model.

Input:

- Objective function $f(x)$
- Number of snakes N
- Number of iterations T
- Parameters (e.g., movement rules, search parameters)

Output:

- Best snake (solution) found

Steps:

1. **Initialization:**
   - Initialize a population of N snakes with random positions x_i in the solution space.
   - Set algorithm parameters, including maximum iterations T, movement rules, and search parameters.
2. **Fitness Evaluation:**
   - For each snake x_i:
     - Compute its fitness $f(x_i)$ using the objective function.
3. **Movement and Adaptation:**
   - For each snake x_i:
     - **Local Search:**
       - Explore the neighbourhood around x_i to identify potentially better solutions.
     - **Global Search:**
       - Update x_i's position based on the best-performing snakes in the population.
       - Calculate the movement direction and velocity considering the following:
         - Fitness of neighbouring snakes.
         - Position of the global best snake.
4. **Update Positions:**
   - Update the position of each snake x_i using the calculated direction and velocity.
5. **Selection and Replacement:**
   - Select the best-performing snakes based on their fitness values.
   - Generate the next generation by:
     - Replacing fewer fit snakes with new random or perturbed snakes.
     - Retaining the best-performing snakes (elitism).
6. **Termination:**
   - Check if stopping criteria are met:
     - Maximum number of iterations T.
     - Convergence (i.e., no significant improvement in fitness).
   - If criteria are not met, return to step 2. Otherwise, proceed to step 7.
7. **Output:**

Return the best snake (solution) found.

Algorithm 1. Algorithm of SOA model.

The FF is a significant factor impacting the efficiency of SOA. The hyper-parameter selection method contains a solution-encoded method to compute the efficiency of the candidate solution. In this study, the SOA finds precision as the primary standard for creating the FF, which can be expressed below.

$$Fitness = \max(P) \tag{31}$$

$$P = \frac{TP}{TP + FP} \tag{32}$$

Here, TP presents the true positive, and FP is the false positive value.

Result analysis and discussion

This section examines the performance validation of the DLBITM-AMD technique using Android malware and Malware detection datasets[38,39]. The dataset contains 15,036 apps with 2 class labels, as denoted in Table 1. A dataset containing feature vectors of 215 attributes was extracted from 15,036 applications (5560 malware apps from the Drebin project and 9476 benign apps). The dataset was employed to progress and assess the multilevel classifier fusion technique for Android malware detection, published in the IEEE Transactions on Cybernetics paper 'Droid Fusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection. The supporting file explains the feature vectors/attributes gained through static code investigation of the Android apps. The suggested technique is simulated using the Python 3.6.5 tool on PC i5-8600k, 250GB SSD, GeForce 1050Ti 4GB, 16GB RAM, and 1 TB HDD. The parameter settings are provided: learning rate: 0.01, activation: ReLU, epoch count: 50, dropout: 0.5, and batch size: 5.

Figure 5 establishes the confusion matrices designed by the DLBITM-AMD method under 80:20 and 70:30 of TRAP/TESP. The results indicate that the DLBITM-AMD technique efficiently identifies and identifies 2 class labels. In the training phase with 80% data, the confusion matrix outputs exhibit a high detection accuracy for benign and malware classes, with the model attaining a true positive rate (TPR) of 98.74% for benign and 98.98% for malware. In comparison, false positive rates (FPR) were subsequently 1.26% and 1.02%. During the testing phase, with 20% data, accuracy improved, with true positive rates of 99.32% for benign and 99.46% for malware, and false positive rates lessened to 0.68% and 0.54%. In the 70% training phase, the model maintained robust performance with true positive rates of 98.68% for benign and 95.99% for malware and false positive rates of 1.32% and 4.01%. Testing with 30% data yielded slightly enhanced outcomes with true positive rates of 98.96% for benign and 97.02% for malware and false positive rates of 1.04% and 2.98%.

Table 2; Fig. 6 show the overall detection of the DLBITM-AMD method under 80%TRAP and 20%TESP. The result is that the DLBITM-AMD approach has properly categorized two classes. With 80%TRAP, the DLBITM-AMD approach gets an average $accu_y$ of 98.62%, $prec_n$ of 98.86%, $reca_l$ of 98.62%, $F_{score}\, of\ 98.74\%,\ and\ MCC$ of 97.48%, respectively. Furthermore, with 20%TESP, the DLBITM-AMD technique obtains an average $accu_y$ of 99.26%, $prec_n$ of 99.39%, $reca_l$ of 99.26%, $F_{score}\, of\ 99.32\%,\ and\ MCC$ of 98.64%, correspondingly.

Table 3; Fig. 7 show the overall recognition of the DLBITM-AMD technique under 70%TRAP and 30%TESP. The result reported that the DLBITM-AMD model has accurately considered two dissimilar classes. With 70%TRAP, the DLBITM-AMD technique attains an average $accu_y$ of 97.69%, $prec_n$ of 97.34%, $reca_l$ of 97.69%, $F_{score}\, of\ 97.51\%\ and\ MCC$ of 95.02%, correspondingly. Besides, with 30%TESP, the DLBITM-AMD methodology attains an average $accu_y$ of 98.24%, $prec_n$ of 97.99%, $reca_l$ of 98.24%, $F_{score}\, of\ 98.11\%,\ and\ MCC$ of 96.23%, correspondingly.

Figure 8 establishes the training and validation accuracy outcomes of the DLBITM-AMD method under 80%TRAP and 20%TESP. The accuracy values are calculated over a range of 0–25 epochs. The figure emphasizes that the training and validation accuracy values display a rising tendency, which informs the capability of the DLBITM-AMD approach with better performance over many iterations. Moreover, the training and validation accuracy rests nearer over the epochs, which designates low, lowest over-fitting and reveals the improved performance of the DLBITM-AMD technique, assuring constant prediction on hidden samples.

Figure 9 shows the training and validation loss graph of the DLBITM-AMD approach under 80%TRAP and 20%TESP. The loss values are calculated throughout 0–25 epochs. The training and validation accuracy values clarify a lessening trend, advising the DLBITM-AMD approach's skill in harmonizing a trade-off between generalization and data fitting. The frequent decrease in loss values also assures the higher performance of the DLBITM-AMD approach and alters the prediction outcomes over time.

In Fig. 10, the precision-recall (PR) curve analysis of the DLBITM-AMD model under 80%TRAP and 20%TESP provides interpretation into its concert by plotting Precision against Recall for every class. The figure shows that the DLBITM-AMD methodology consistently achieves upgraded PR values across dissimilar class labels, signifying its capacity to preserve a significant part of true positive forecasts among every positive prediction (precision) while taking a massive quantity of actual positives (recall). The stable augmentation in PR results between every class represents the efficiency of the DLBITM-AMD approach in the classification procedure.

In Fig. 11, the ROC curve of the DLBITM-AMD methodology is studied. The outcomes suggest that the DLBITM-AMD approach under 80%TRAP and 20%TESP gets superior ROC results over every class, representing a substantial ability to discern the classes. This consistent trend of enhanced ROC values over numerous class labels indicates the proficient performance of the DLBITM-AMD technique in predicting classes, emphasizing the robust nature of the classification procedure.

The comparative analysis of the DLBITM-AMD approach with existing models under the Android malware dataset is presented in Table 4; Fig. 12[1,3,40]. The experimental results clearly state that the DLBITM-AMD technique portrayed superior performance improvements. Based on $accu_y$, the DLBITM-AMD technique achieves a larger $accu_y$ of 99.26%, while the RHSODL-AMD, AAMD-OELAC, GBWODL-AMC, DBN, J48,

Classes	No. of apps
Benign	9476
Malware	5560
Total apps	15,036

Table 1. Details on dataset.

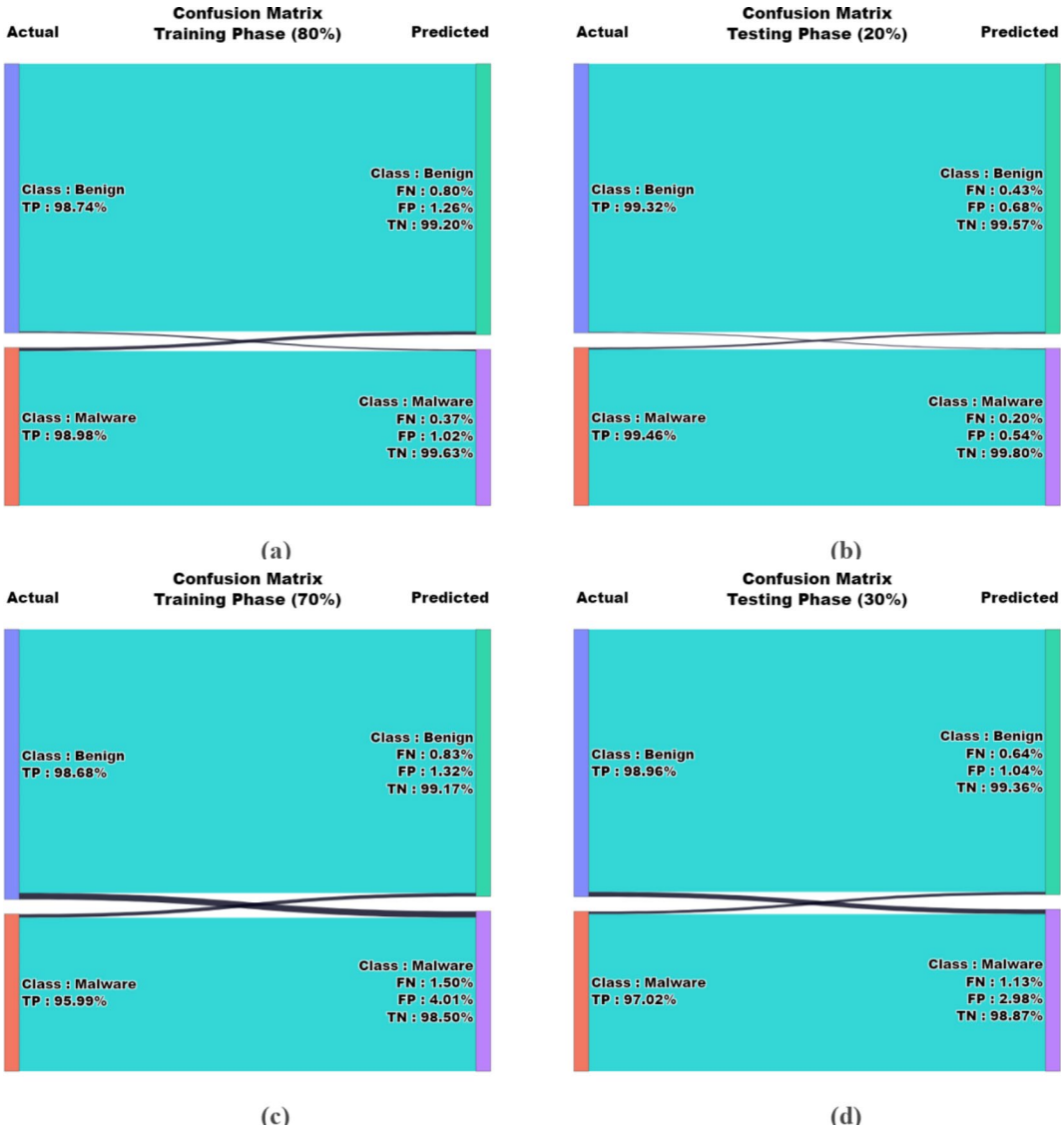

Fig. 5. Confusion matrices of DLBITM-AMD technique (**a**,**b**) 80%TRAP and 20%TESP and (**c**,**d**) 70%TRAP and 30%TESP.

Class	$Accu_y$	$Prec_n$	$Reca_l$	F_{score}	MCC
TRAP (80%)					
Benign	99.41	98.74	99.41	99.07	97.48
Malware	97.84	98.98	97.84	98.41	97.48
Average	98.62	98.86	98.62	98.74	97.48
TESP (20%)					
Benign	99.68	99.32	99.68	99.50	98.64
Malware	98.83	99.46	98.83	99.14	98.64
Average	99.26	99.39	99.26	99.32	98.64

Table 2. Classifier outcome of DLBITM-AMD technique under 80%TRAP and 20%TESP.

Naïve Bayes, and Linear-SVM techniques reached lesser $accu_y$ of 99.05%, 98.93%, 98.95%, 96.81%, 94.85%, 96.64%, and 94.45%, respectively. Similarly, based on $prec_n$, the DLBITM-AMD methodology portrayed a greater $prec_n$ of 99.39%. In contrast, the RHSODL-AMD, AAMD-OELAC, GBWODL-AMC, DBN, J48, Naïve Bayes, and Linear-SVM approaches have lesser $prec_n$ of 99.02%, 99.15%, 97.35%, 97.46%, 94.26%, 97.42%, and 94.5%, respectively. Likewise, based on $reca_l$, the DLBITM-AMD technique attains a greater value of 99.26%,

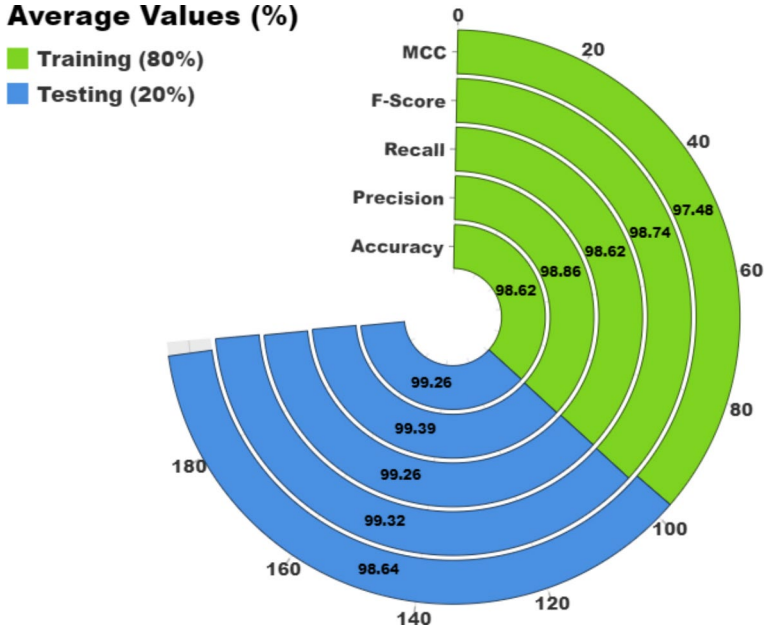

Fig. 6. Average of DLBITM-AMD technique under 80%TRAP and 20%TESP.

Class	$Accu_y$	$Prec_n$	$Reca_l$	F_{score}	MCC
TRAP (70%)					
Benign	97.63	98.68	97.63	98.15	95.02
Malware	97.75	95.99	97.75	96.87	95.02
Average	97.69	97.34	97.69	97.51	95.02
TESP (30%)					
Benign	98.19	98.96	98.19	98.58	96.23
Malware	98.28	97.02	98.28	97.65	96.23
Average	98.24	97.99	98.24	98.11	96.23

Table 3. Classifier outcome of DLBITM-AMD technique under 70%TRAP and 30%TESP.

whereas the RHSODL-AMD, AAMD-OELAC, GBWODL-AMC, DBN, J48, Naïve Bayes, and Linear-SVM models attained minimum $reca_l$ of 99.05%, 98.93%, 99.04%, 96.82%, 94.92%, 96.74%, and 95.42%, subsequently. Lastly, based on F_{score}, the DLBITM-AMD method has a larger F_{score} of 99.32%, while the RHSODL-AMD, AAMD-OELAC, GBWODL-AMC, DBN, J48, Naïve Bayes, and Linear-SVM techniques demonstrated lesser F_{score} values of 99.03%, 99.04%, 97.35%, 97.99%, 94.09%, 97.72%, and 94.76%, respectively.

The comparative analysis of the DLBITM-AMD approach with existing methods under the Malware detection dataset is given in Table 5; Fig. 13[41]. The investigational outcomes confirm that the DLBITM-AMD technique significantly outperforms other models. Based on $accu_y$, the DLBITM-AMD technique has a larger $accu_y$ of 99.17%. In contrast, the LSTM Classifier, KNN, RF, CNN Encoder-RNN, DNN, CNN Classifier, and Dense Model techniques attained lesser $accu_y$ of 93.00%, 98.90%, 91.97%, 96.00%, 94.00%, 98.00%, and 98.38%, respectively. Similarly, based on $prec_n$, the DLBITM-AMD methodology portrayed a greater $prec_n$ of 99.21%. In contrast, the LSTM Classifier, KNN, RF, CNN Encoder-RNN, DNN, CNN Classifier, and Dense Model approaches have lesser $prec_n$ of 98.10%, 96.07%, 93.85%, 97.23%, 95.24%, 97.81%, and 98.77%, respectively. Likewise, based on $reca_l$, the DLBITM-AMD methodology illustrated a superior value of 98.91%. In contrast, the LSTM Classifier, KNN, RF, CNN Encoder-RNN, DNN, CNN Classifier, and Dense Model methods attained a minimum $reca_l$ of 98.40%, 94.30%, 93.10%, 96.56%, 93.55%, 96.93%, and 97.03%, subsequently. Lastly, based on F_{score}, the DLBITM-AMD method has a larger F_{score} of 98.91%. At the same time, the LSTM Classifier, KNN, RF, CNN Encoder-RNN, DNN, CNN Classifier, and Dense Model techniques demonstrated lesser F_{score} values of 98.22%, 97.02%, 96.41%, 94.64%, 98.71%, 96.71%, and 93.29%, respectively.

In Table 6; Fig. 14, the comparative outcomes of the DLBITM-AMD technique are stated in terms of time cost (TC). The results suggest that the DLBITM-AMD method gets enhanced performance. Based on TC, the DLBITM-AMD method shows lesser TC of 0.82s while the RHSODL-AMD, AAMD-OELAC, GBWODL-AMC, DBN, J48, Naïve Bayes, and Linear-SVM approaches have attained greater TC values of 1.62 s, 2.87 s, 3.40 s, 3.41 s, 3.43 s, 2.28 s, and 1.68 s, correspondingly.

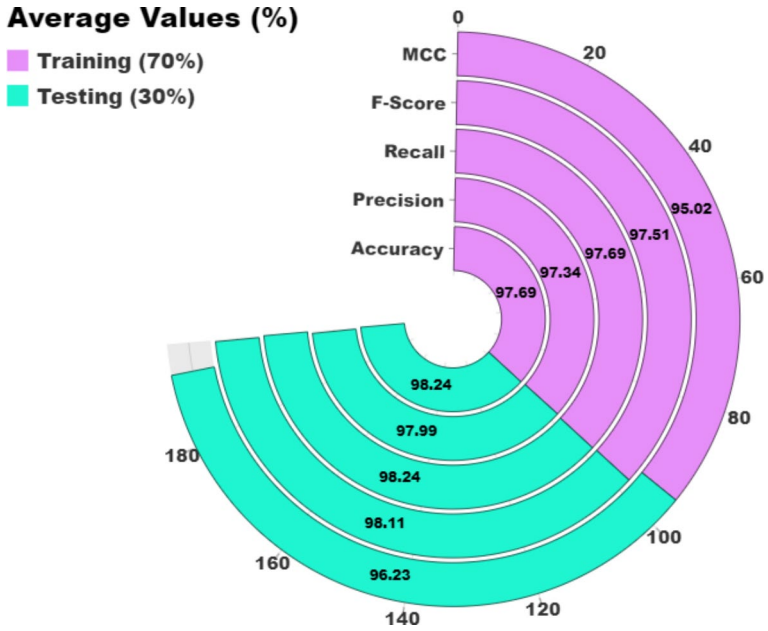

Fig. 7. Average of DLBITM-AMD technique under 70%TRAP and 30%TESP.

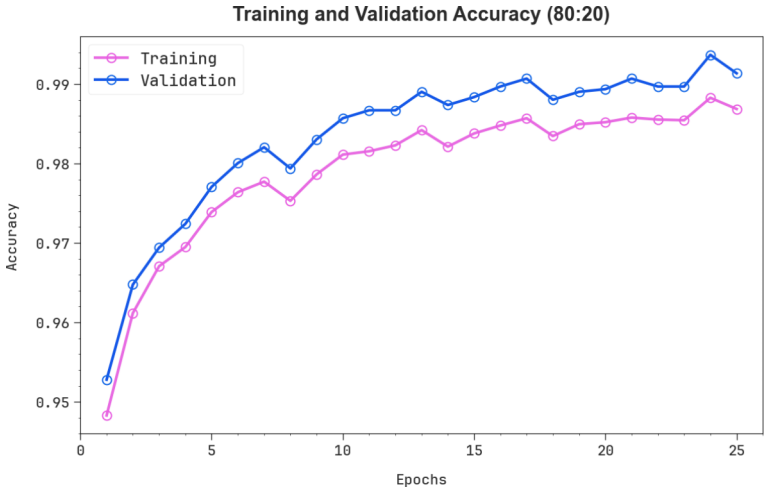

Fig. 8. $Accu_y$ curve of DLBITM-AMD technique under 80%TRAP and 20%TESP

In Table 7; Fig. 15, the comparative outcomes of the DLBITM-AMD technique are illustrated in terms of TC. The outcomes suggest that the DLBITM-AMD method gets enhanced performance. Based on TC, the DLBITM-AMD method portrays a lesser TC of 4.61 s while the LSTM Classifier, KNN, RF, CNN Encoder-RNN, DNN, CNN Classifier, and Dense Model methods attained greater TC values of 9.67 s, 9.72 s, 7.77 s, 8.15 s, 9.25 s, 8.63 s, and 6.82 s, subsequently.

Conclusion

In this manuscript, the DLBITM-AMD approach is proposed for IoVs. The main aim of the presented DLBITM-AMD approach is to detect Android malware effectually and accurately. The DLBITM-AMD method performs a Z-score normalization process to convert the raw data into a standard form. Then, the DLBITM-AMD approach utilizes the BGWO model to select optimum feature subsets. An improved transformer is integrated with the RNN model and softmax to enhance classification for Android malware recognition. Finally, the SOA method is employed to select the optimum parameter of the classification method. An extensive experiment of the DLBITM-AMD method is accomplished on a benchmark dataset. The performance validation of the DLBITM-AMD technique portrayed a superior accuracy value of 99.26% over existing Android malware recognition models. The DLBITM-AMD technique, while comprehensive, has few limitations. Z-score normalization may not address all data anomalies, potentially affecting the technique's performance in highly variable datasets. The BGWO-based

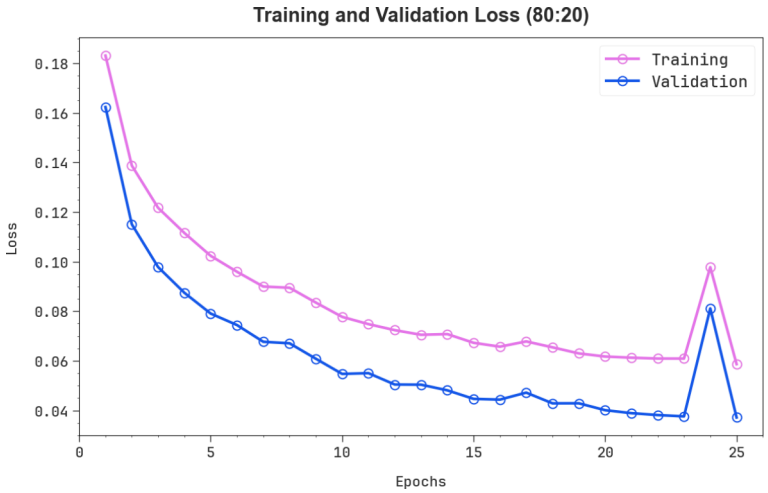

Fig. 9. Loss curve of DLBITM-AMD technique under 80%TRAP and 20%TESP.

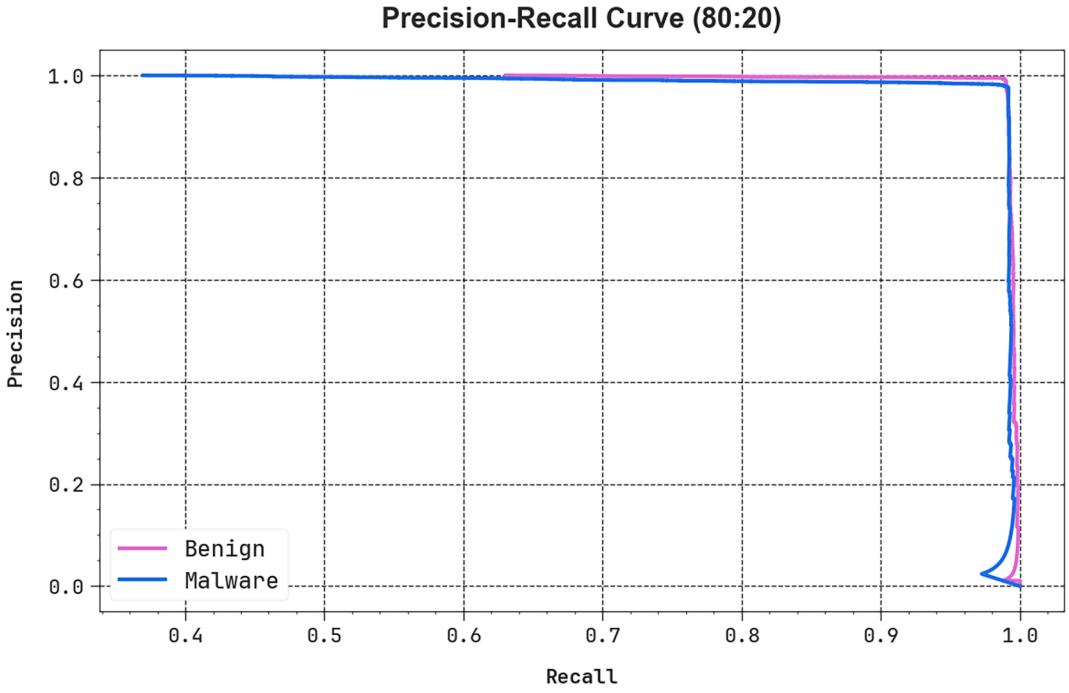

Fig. 10. PR curve of DLBITM-AMD technique under 80%TRAP and 20%TESP.

FS may encounter difficulty capturing all relevant features, specifically in dynamic environments. The Enhanced transformer-RNN-based classification may face challenges with scalability and high computational demands. The SOA-based hyperparameter tuning may be intricate and time-consuming. Furthermore, addressing the model's adaptability to diverse data types and enhancing generalization capabilities are significant areas for future development. Moreover, enhancing model interpretability and reducing computational demands will be substantial for broader adoption and practical use. Future work should concentrate on enhancing the robustness of normalization models, exploring alternative FS techniques, optimizing computational efficiency, and simplifying hyperparameter tuning processes to improve the comprehensive efficiency and adaptability of the DLBITM-AMD technique. Future studies should concentrate on improving the robustness of the model against adversarial attacks and noisy data to confirm reliability in practical scenarios. Moreover, incorporating advanced preprocessing and augmentation methods could enhance feature extraction and overall accuracy. Furthermore, exploring the adaptation of models for cross-domain applicability will confirm that these methodologies remain effectual across a variety of contexts and data types.

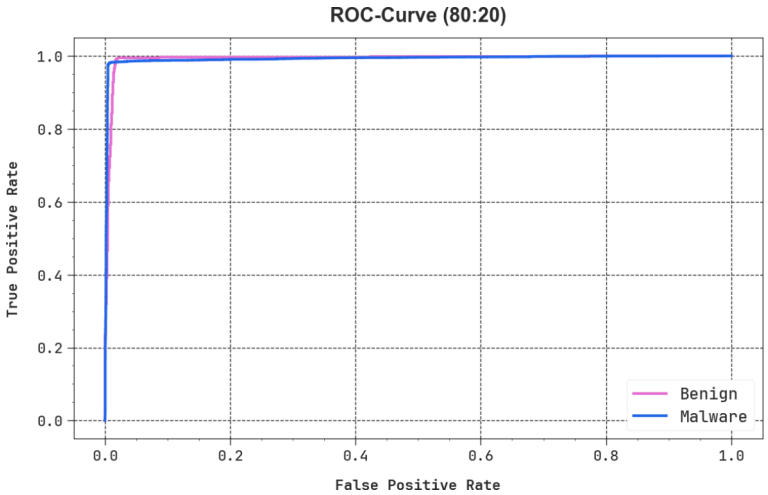

Fig. 11. ROC curve of DLBITM-AMD technique under 80%TRAP and 20%TESP.

Methods	$Accu_y$	$Prec_n$	$Reca_l$	F_{score}
DLBITM-AMD	99.26	99.39	99.26	99.32
RHSODL-AMD	99.05	99.02	99.05	99.03
AAMD-OELAC	98.93	99.15	98.93	99.04
GBWODL-AMC	98.95	97.35	99.04	97.35
DBN	96.81	97.46	96.82	97.99
J48	94.85	94.26	94.92	94.09
Naïve–Bayes	96.64	97.42	96.74	97.72
Linear-SVM	94.45	94.50	95.42	94.76

Table 4. Comparative analysis of the DLBITM-AMD method with recent models under the Android malware dataset.

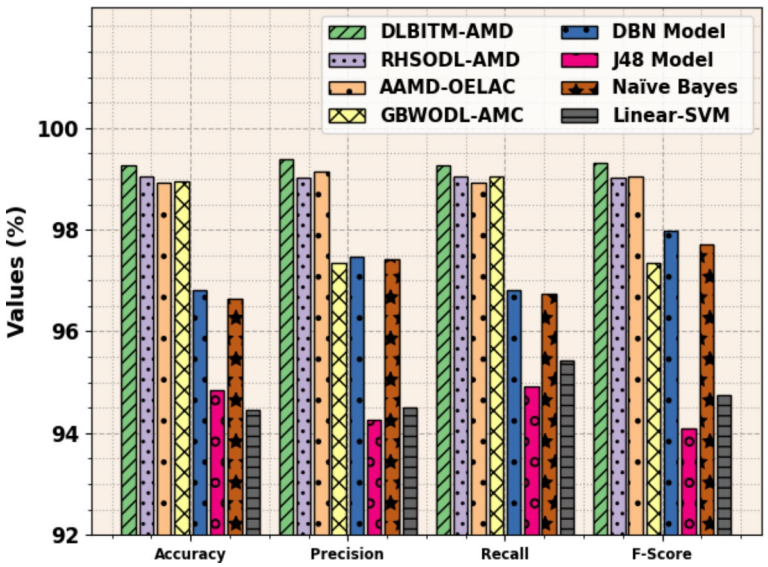

Fig. 12. Comparative analysis of DLBITM-AMD technique with recent models under Android malware dataset.

Methods	$Accu_y$	$Prec_n$	$Reca_l$	F_{score}
LSTM classifier	93.00	98.10	98.40	98.22
KNN	98.90	96.07	94.30	97.02
RF	91.97	93.85	93.10	96.41
CNN encoder-RNN	96.00	97.23	96.56	94.64
DNN	94.00	95.24	93.55	98.71
CNN classifier	98.00	97.81	96.93	96.71
Dense model	98.38	98.77	97.03	93.29
DLBITM-AMD	99.17	99.21	99.07	98.91

Table 5. Comparative analysis of the DLBITM-AMD method with recent models under Malware detection dataset.

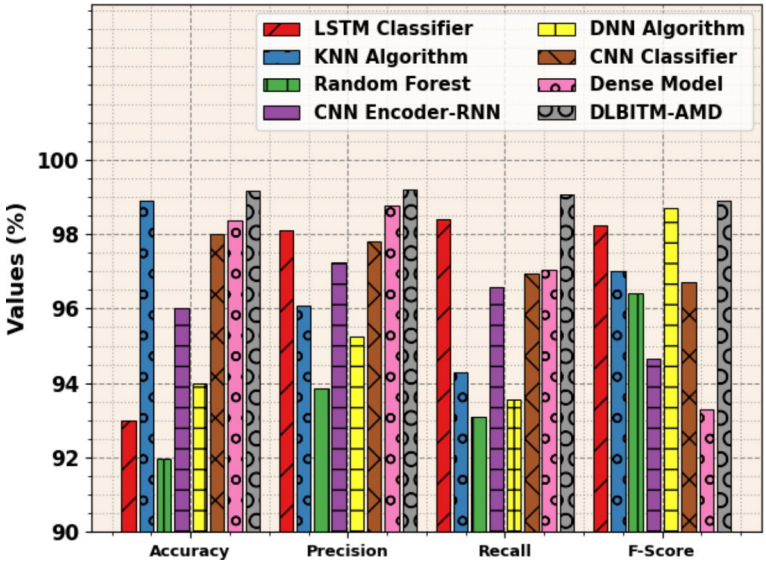

Fig. 13. Comparative analysis of DLBITM-AMD technique with recent models under Malware detection dataset.

Methods	TC (s)
DLBITM-AMD	0.82
RHSODL-AMD	1.62
AAMD-OELAC	2.87
GBWODL-AMC	3.40
DBN	3.41
J48	3.43
Naïve Bayes	2.28
Linear-SVM	1.68

Table 6. TC of DLBITM-AMD technique with recent models under android malware dataset.

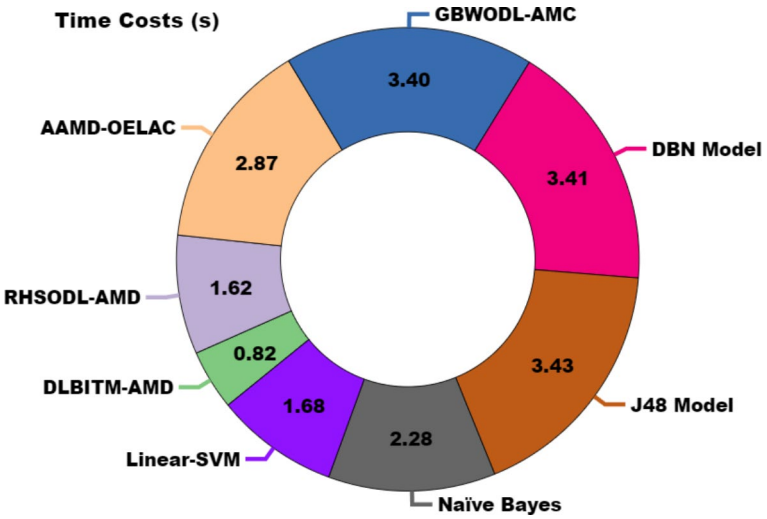

Fig. 14. TC of DLBITM-AMD technique with recent models under android malware dataset.

Methods	TC (s)
LSTM classifier	9.67
KNN	9.72
RF	7.77
CNN encoder-RNN	8.15
DNN	9.25
CNN classifier	8.63
Dense model	6.82
DLBITM-AMD	4.61

Table 7. TC of DLBITM-AMD technique with recent methods under malware detection dataset.

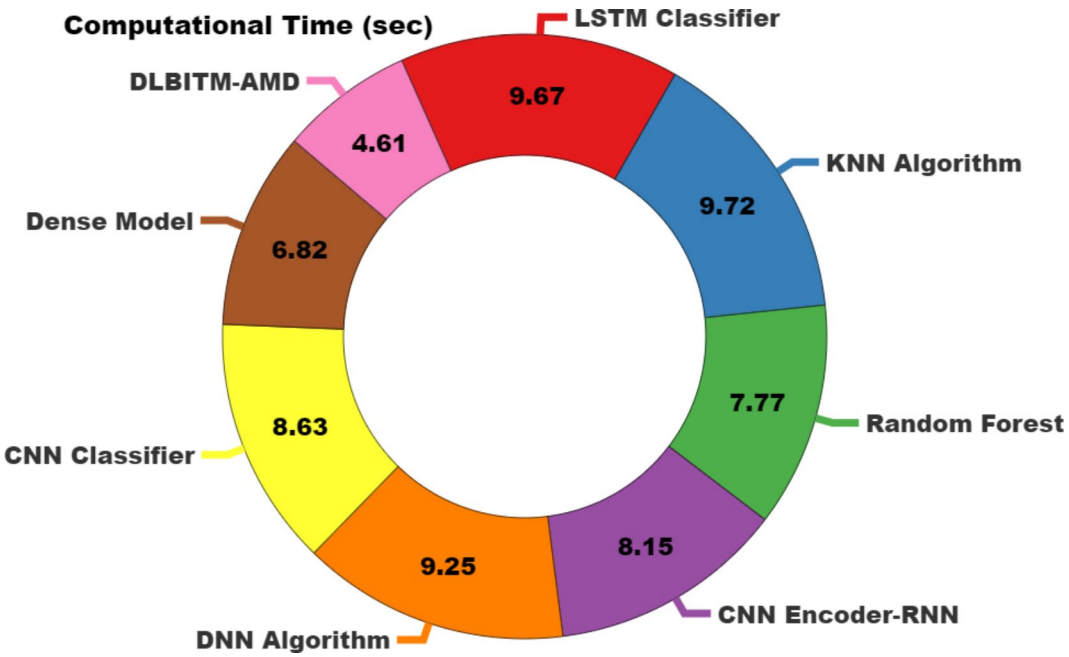

Fig. 15. TC of DLBITM-AMD technique with recent models under malware detection dataset.

Data availability

The datasets used and analyzed during the current study available from the corresponding author on reasonable request.

Received: 16 July 2024; Accepted: 23 September 2024
Published online: 24 October 2024

References

1. Aldehim, G. et al. Gauss-Mapping Black Widow optimization with Deep Extreme Learning Machine for Android Malware classification model. *IEEE Access.* **11**, 87062–87070. https://doi.org/10.1109/ACCESS.2023.3285289 (2023).
2. Smmarwar, S. K., Gupta, G. P. & Kumar, S. A hybrid feature selection approach-based Android malware detection framework using machine learning techniques. In *Cyber Security, Privacy and Networking: Proceedings of ICSPN 2021* 347–356 (Springer Nature, 2022). https://doi.org/10.1007/978-981-16-8664-1_30.
3. Alamro, H. et al. Automated android Malware Detection using Optimal Ensemble Learning Approach for Cybersecurity. *IEEE Access.* **11**, 72509–72517. https://doi.org/10.1109/ACCESS.2023.3294263 (2023).
4. Ficco, M. Detecting IoT Malware by Markov Chain Behavioral Models, In *IEEE International Conference on Cloud Engineering (IC2E)*, Prague, Czech Republic, 229–234 (2019). https://doi.org/10.1109/IC2E.2019.00037.
5. Sapalo Sicato, J. C., Sharma, P. K., Loia, V. & Park, J. H. VPNFilter malware analysis on cyber threat in smart home network. *Appl. Sci.* **9** (13), 2763 (2019).
6. Liu, K. et al. A review of android malware detection approaches based on machine learning. *IEEE Access.* **8**, 124579–124607 (2020).
7. Inayat, U. et al. Learning-based methods for cyber attacks detection in IoT systems: A survey on methods, analysis. *Electronics* **11**(9), 1502 (2022).
8. Qiu, J. et al. A survey of android malware detection with deep neural models. *ACM Comput. Surv. (CSUR).* **53** (6), 1–36 (2020).
9. Zhao, S., Li, S., Qi, L. & Xu, L. D. Computational intelligence enabled Cybersecurity for the internet of things. *IEEE Trans. Emerg. Top. Comput. Intell.* **4** (5), 666–674 (2020).
10. Dovom, E. M. et al. Fuzzy pattern tree for edge malware detection and categorization in IoT. *J. Syst. Architect.* **97**, 1–7 (2019).
11. Zhao, Y. & Liu, Y. *Malware Detection Based on Optimized Deep Learning in Data-driven Mode* (2024).
12. de Oliveira, A. S. & Sassi, R. J. Chimera: an android malware detection method based on multimodal deep learning and hybrid analysis. *Authorea Preprints*. https://doi.org/10.36227/techrxiv.13359767.v1 (2023).
13. Almutlaq, S., Derhab, A., Hassan, M. M. & Kaur, K. Two-stage intrusion detection system in intelligent transportation systems using rule extraction methods from deep neural networks. *IEEE Trans. Intell. Transp. Syst.* **24** (12), 15687–15701 (2022).
14. Bakır, H. VoteDroid: a new ensemble voting classifier for malware detection based on fine-tuned deep learning models. *Multimedia Tool. Appl.* 1–22. https://doi.org/10.1007/s11042-024-19390-7 (2024).
15. Luo, F. et al. A multilayer intrusion detection system for SOME/IP-based in-vehicle network. *Sensors*. **23** (9), 4376 (2023).
16. Al-Hawawreh, M. & Moustafa, N. Explainable deep learning for attack intelligence and combating cyber-physical attacks. *Ad Hoc Netw.* **153**, 103329 (2024).
17. Abdullah, M. A. et al. HCL-Classifier: CNN and LSTM-based hybrid malware classifier for the internet of things (IoT). *Future Gener. Comput. Syst.* **142**, 41–58 (2023).
18. Pravin, A., Prem Jacob, T. & Raja Kumar, R. Circle search optimization-based deep Q-learning network for intrusion detection system in cloud environment. *IETE J. Res.* **1–15**https://doi.org/10.1080/03772063.2024.2351556 (2024).
19. Alguliyev, R., Aliguliyev, R. & Sukhostat, L. Radon transform based malware classification in cyber-physical system using deep learning. *Results Control Optim.* **14**, 100382 (2024).
20. Jo, J., Cho, J. & Moon, J. A malware detection and extraction method for the related information using the ViT attention mechanism on android operating system. *Appl. Sci.* **13** (11), 6839 (2023).
21. Ahmad, I., Wan, Z., Ahmad, A. & Ullah, S. S. A hybrid optimization model for efficient detection and classification of Malware in the internet of things. *Mathematics*. **12** (10), 1437 (2024).
22. Zhan, D. et al. An adversarial robust behavior sequence anomaly detection approach based on critical behavior unit learning. *IEEE Trans. Comput.* **72** (11), 3286–3299. https://doi.org/10.1109/TC.2023.3292001 (2023).
23. Malini, P. & Kavitha, K. R. An efficient deep learning mechanisms for IoT/Non-IoT devices classification and attack detection in SDN-enabled smart environment. *Comput. Secur.* **141**, 103818 (2024).
24. Wang, H. et al. An intelligent digital twin method based on spatio-temporal feature fusion for IoT attack behavior identification. *IEEE J. Sel. Areas Commun.* **41**, 3561–3572. https://doi.org/10.1109/JSAC.2023.3310091 (2023).
25. Egitmen, A., Gokhan Yavuz, A. & Yavuz, S. TRConv: multi-platform malware classification via target regulated convolutions. *IEEE Access.* **12**, 71492–71504. https://doi.org/10.1109/ACCESS.2024.3401627 (2024).
26. Zhao, R. et al. A novel self-supervised framework based on masked autoencoder for traffic classification. *IEEE/ACM Trans. Netw.*. **32** (3), 2012–2025. https://doi.org/10.1109/TNET.2023.3335253 (2024).
27. Mai, J. et al. Anomaly detection method for vehicular network based on collaborative deep support vector data description. *Phys. Commun.* **56**, 101940 (2023).
28. Sun, H., Chen, M., Weng, J., Liu, Z. & Geng, G. Anomaly detection for in-vehicle network using CNN-LSTM with attention mechanism. *IEEE Trans. Veh. Technol.* **70** (10), 10880–10893 (2021).
29. Sun, H. et al. CCID-CAN: Cross-chain Intrusion detection on CAN Bus for autonomous vehicles. *IEEE Internet Things J.* (2024).
30. Kishore, C. R. & Behera, H. S. Malware attack detection in vehicle cyber physical system for planning and control using deep learning. In *Machine Learning for Cyber Physical System: Advances and Challenges* 167–193 (Springer Nature, 2024).
31. Maray, M. et al. Intelligent pattern recognition using equilibrium optimizer with deep learning model for android malware detection. *IEEE Access.* (2024).
32. Haouas, I., Attia, M., Hamel, L., Graiet, M. & Gaaloul, W. Efficient deep learning method for detection of malware attacks in internet of things networks. In *Asian Conference on Intelligent Information and Database Systems* 15–26 (Springer Nature, 2024).
33. Prihanditya, H. A. The implementation of z-score normalization, boosting techniques to increase the accuracy of the c4, 5 algorithm in diagnosing chronic kidney disease. *J. Soft Comput. Explor.* **1** (1), 63–69 (2020).
34. Bilal, A. et al. Advanced CKD detection through optimized metaheuristic modeling in healthcare informatics. *Sci. Rep.* **14** (1), 12601 (2024).
35. Hua, G., Sun, Y. & Li, W. Hybrid load prediction model of 5G base station based on time series decomposition and GRU network with parameter optimization. *IET Gener. Transm. Distrib.* **18** (8), 1548–1558 (2024).
36. Luo, A. et al. An improved transformer-based model for long-term 4D trajectory prediction in civil aviation. *IET Intel. Transport Syst.*https://doi.org/10.1049/itr2.12530 (2024).
37. Kong, L. G. et al. Optimize photovoltaic MPPT with improved snake algorithm. *Energy Rep.* **11**, 5033–5045 (2024).
38. https://www.kaggle.com/datasets/shashwatwork/android-malware-dataset-for-machine-learning
39. https://www.kaggle.com/datasets/nsaravana/malware-detection?select=Malware+dataset.csv
40. Albakri, A., Alhayan, F., Alturki, N., Ahamed, S. & Shamsudheen, S. Metaheuristics with deep learning model for cybersecurity and android malware detection and classification. *Appl. Sci.* **13** (4), 2172 (2023).

41. Alomari, E. S. et al. Malware detection using deep learning and correlation-based feature selection. *Symmetry*. **15** (1), 123 (2023).

Acknowledgements
The authors thank the Deanship of Research and Graduate Studies at King Khalid University for funding this work through a Large Research Project under grant number RGP2/297/45.

Author contributions
Conceptualization, data curation and formal analysis, investigation and methodology, project administration and resources: supervision, validation and visualization, writing—original draft, review and editing: N.A.

Declarations

Competing interests
The author declares no competing interests.

Ethics approval
This article contains no studies with human participants performed by any authors.

Additional information
Correspondence and requests for materials should be addressed to N.A.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.