Received: March 14, 2024. Revised: May 27, 2024.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Hybrid Artificial Fish Swarm Optimization with Deep Learning-Driven Cloud Assisted Cyberattack Detection

Ahmed Al-Khayyat[1,2,3] Mahmood Anees Ahmed[4] Ahmad Taher Azar[5, 6, 7]
Zeeshan Haider[5, 6] Ibraheem Kasim Ibraheem[8, 9]*

[1]College of Technical Engineering, the Islamic University, Najaf, Iraq
[2]College of Technical Engineering, the Islamic University of Al Diwaniyah, Al Diwaniyah, Iraq
[3]College of Technical Engineering, the Islamic University of Babylon, Babylon, Iraq
[4]Medical Instrumentation Techniques Engineering Department,
College of Medical Techniques. Al-Farahidi University, Baghdad 10001, Iraq
[5]College of Computer and Information Sciences, Prince Sultan University, Riyadh, Saudi Arabia
[6]Automated Systems and Soft Computing Lab (ASSCL), Prince Sultan University, Riyadh, Saudi Arabia
[7]Faculty of Computers and Artificial Intelligence, Benha University, Benha, Egypt
[8]Department of Electrical Engineering, College of Engineering, University of Baghdad, Baghdad 10001, Iraq
[9]Department of Electronics and Communication Engineering,
College of Engineering, Uruk University, Baghdad, Iraq
* Corresponding author's Email: ibraheemki@coeng.uobaghdad.edu.iq

Abstract: In the evolving landscape of cloud computing and the Internet of Things (IoT), the Android Operating System (AOS) has emerged as a focal point for cybersecurity efforts, particularly due to its vulnerability to a wide array of cyberattacks. These threats, which include financial loss, privacy breaches, unauthorized access, data integrity compromises, and denial of services (DoS), have accentuated the need for advanced malware detection solutions. This study introduces a pioneering cloud-enabled Hybrid Artificial Fish Swarm Optimization with Deep Learning-Driven Malware Detection (HAFSO-DLMD) technique for Android devices, aiming to enhance the precision of malware identification through deep learning models. The HAFSO-DLMD technique preprocesses the bytecodes of Android applications' classes.dex files for input into a Deep Sparse Autoencoder (DSAE) represent a significant innovation in the field. By employing the HAFSO algorithm for optimal hyperparameter tuning, we have demonstrated a substantial improvement in the detection rate of the DSAE model. Our comprehensive experimental evaluation on an Android APK dataset comprising 16,000 samples has underscored the HAFSO-DLMD technique's superior performance, achieving accuracy, precision, recall, and F_{score} of 99.17%. These results significantly outperform other contemporary approaches, thereby establishing the HAFSO-DLMD method as a potent tool in bolstering Android's cybersecurity infrastructure within cloud environments.

Keywords: Cybersecurity, Cloud computing, Deep learning, Android malware, Internet of things, Fish swarm optimization algorithm.

1. Introduction

By leveraging massive datasets related to the malware and computational resources, cloud-based services offer enhanced response times and numerous advantages, including resource pooling, scalability, on-demand service, and improved network access. However, there are some issues as well in the case of cloud features, such as network unavailability, security issues, etc. Nowadays, the scope of the Internet of Things (IoT) is expanding with the advent of various applications like smart homes, healthcare, smart shopping, and intelligent agriculture [1]. Many gadgets in a shared IoT network depend on the Android platform because of hardware support,

flexibility, and robustness, which are vital for sensor interfaces. The distinct kinds of IoT gadgets offer various eminent services relevant to controlling, sensing, and monitoring tasks [2].

Android is the prominent platform for IoT gadgets, increasing the number of applications accessible in the market, specifically Android applications [3]. Opponents build different kinds of malicious applications. For instance, using open ports in which the attacker had control over the device of the user by opening one of the ports on the gadgets [4]. This method permits the attacker to access all of the device's resources remotely without demanding permission from the owner. Hence, hackers use Android applications to break the security of the device, which permits them to access delicate data like contact information, photos, and the device's location [5]. Fig. 1 represents the process involved in cloud-assisted malware detection.

The current studies emphasize malware detection for Android devices [6]. Still, Android devices have been targeted by hackers because of the widespread use of the Android platform in IoT gadgets. Conventional malware recognition techniques depend mainly on accumulating signature libraries and human interference by malware analysts. Hence, it is tough to adapt to the explosive development of Android malware [7]. With the data accumulation and continuous enhancement of computational power, machine learning (ML) technology was broadly implemented in these three kinds of recognition approaches, offering another viewpoint for automatic and efficient Android malware detection. The ML-related Android malware detection approaches primarily include the following four steps: First, data were constructed by gathering malicious and benign Android applications [8].

Secondly, feature engineering can be executed to extract features to describe Android applications. Then, ML methods were trained to detect malware.

At last, the trained model's performance can be assessed by forecasting test samples [9]. Also, many previous studies modeled for Android malware recognition depend on the limited kinds of feature selection for detecting malware. Also, many prevailing malware detection approaches cannot be directly used for IoT gadgets because of their limited resources, such as cost, memory, processing abilities, and computation complexity [10]. Numerous methods are developed to select features of malware, for example, the information gain method, but they are relevant to limited types of features for malware detection.

This study introduces a Hybrid Artificial Fish Swarm Optimization with Deep Learning - Driven Malware Detection (HAFSO-DLMD) technique for Android IoT devices. The goal of the HAFSO-DLMD technique lies in the proper identification of Android malware using the DL model. The HAFSO-DLMD technique initially preprocesses the actual bytecodes of the classes.dex file of Android application as an input to the DL model. In the presented HAFSO-DLMD technique, the deep sparse autoencoder (DSAE) model is applied for malware detection purposes. The work's originality is demonstrated by the application of the HAFSO technique to optimize the hyperparameter and increase the DSAE model's detection rate. The experimental result assessment of the HAFSO-

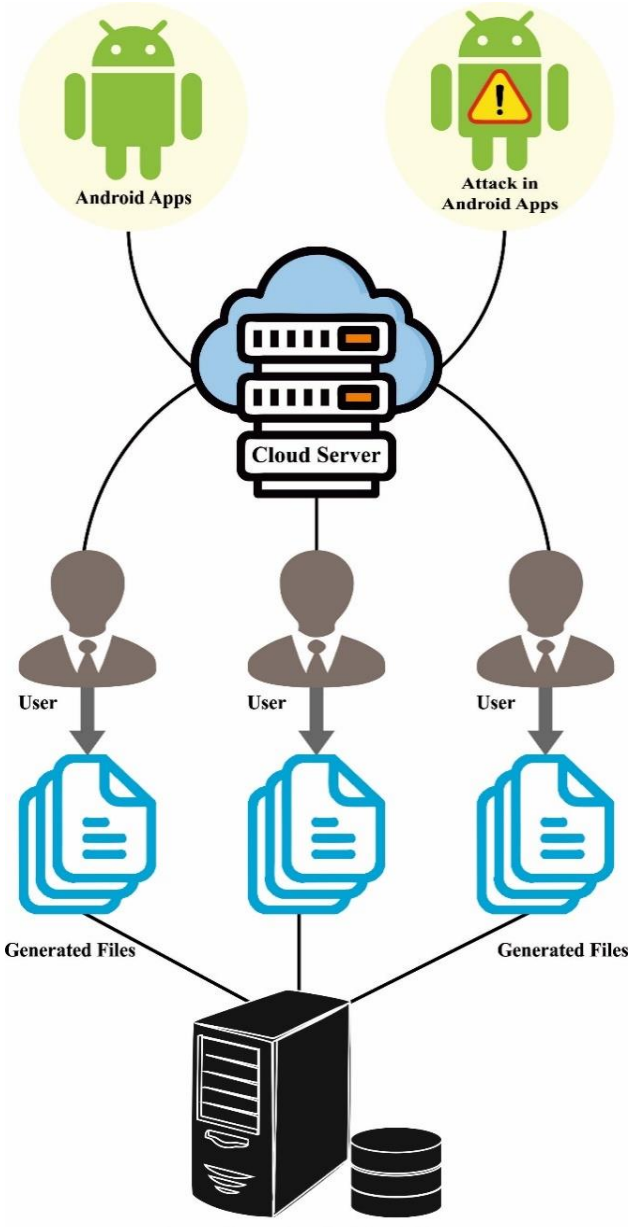


Figure. 1 Method for Identifying Attacks in Android Applications using Cloud-Based Server Analysis

DLMD technique takes place on the Android APK dataset.

As we navigate through the challenges posed by the evolving cybersecurity landscape, this paper introduces an innovative approach that harnesses the synergy between Hybrid Artificial Fish Swarm Optimization (HAFSO) and Deep Learning-Driven Malware Detection (DLMD) to enhance malware detection capabilities within Android Operating Systems. The novelty of this research lies in several key areas:

1. **The adoption of a pioneering hybrid approach** that integrates HAFSO with DLMD, targeting the precise identification of malware through advanced deep learning models, sets a new standard in the field of cybersecurity, specifically tailored for cloud environments and Android platforms.
2. **Utilization of the Deep Sparse Autoencoder (DSAE)**, a cornerstone of our methodology, enables the effective processing and analysis of Android applications bytecode. This advanced model is instrumental in discerning benign from malicious app patterns, significantly bolstering our detection capabilities.
3. **Optimization of hyperparameters through the HAFSO algorithm** emerges as a critical component of our strategy, ensuring that our deep learning models operate at peak efficiency. This optimization process is pivotal in surpassing the detection accuracy of existing methods.
4. **An extensive experimental evaluation** underscores the robustness of our approach. By rigorously testing the HAFSO-DLMD technique on a comprehensive dataset, we demonstrate its superiority in accuracy, precision, recall, and F-score compared to contemporary approaches.
5. **A detailed comparative analysis** further positions the HAFSO-DLMD technique as a leading solution in malware detection. This comparative insight not only highlights the effectiveness of our method but also situates our contributions within the broader cybersecurity research landscape.
6. **The practical implications of our research** extend to enhancing Android's cybersecurity infrastructure, offering significant advancements for professionals, developers, and researchers dedicated to safeguarding against cyber threats.
7. **Future work and continued research** are discussed, laying the groundwork for further advancements in malware detection technologies. Our suggestions for future research endeavors underscore our commitment to ongoing innovation in the cybersecurity domain.

The subsequent sections of this paper delve into the methodology, experimental setup, results, and comparative analysis in detail, illustrating the comprehensive nature of our study and its contributions to the field of cybersecurity.

The remainder of this paper is structured as follows: Section 2 presents a detailed review of related works, establishing the context for our research. The problem statement is stated in Section 3. Section 4 elaborates on the methodology, including the Hybrid Artificial Fish Swarm Optimization and Deep Learning-Driven Malware Detection technique. Section 5 describes the experimental setup and dataset used for evaluation. In Section 6, we present our results and conduct a comprehensive analysis, comparing our approach with existing methods. Section 7 discusses the implications of our findings and suggests directions for future research. Finally, Section 8 concludes the paper, summarizing the key contributions and potential impacts on the field of cybersecurity.

2. Literature review

Dhabal and Gupta [11] presented a new Android malware detection structure utilizing hybrid DL approaches. In the proposed structure, initially, the preprocessing Step was utilized to have an enhanced feature subset. For FS, this study has used gain data and Pearson correlation coefficient methods where the k-best feature was chosen through the gain information method. For detecting malware, optimized feature-based data was utilized to train the presented hybrid of merged sparse auto-encoder (MSAE) and bidirectional LSTM (BiLSTM) with the softmax DL method. Kim and colleagues [12] developed MAPAS, a malware detection system designed to achieve greater accuracy and efficient computational resource utilization. MAPAS evaluates the behavioral patterns of harmful applications by scrutinizing API call graphs with the help of CNN. Rather than using a CNN-generated model, MAPAS utilizes CNN to identify characteristic attributes of malware's API call graphs.

Fallah and Bidgoly [13] proposed a technique that depends on LSTM for malware detection, which can distinguish benign and malware samples and identify the unseen and new families of malware. This is the first time that a traffic dataset was devised as a series of flows, and a sequential related DL method was used. In [14], the author developed 8 Android malware detection techniques relevant to DNN and ML and inspected their robustness contrary to adversarial assaults. For this purpose, the author created a new version of malware using RF, which

would be misclassified as benevolent by prevailing Android malware detection methods. The author modeled 2 new attack approaches called multiple policy attacks utilizing RL and single policy assault for grey-box and white-box scenarios correspondingly.

The study of [15] devised a DL structure utilizing network traffic features for detecting Android malware. Generally, ML techniques required data preprocessing, but such preprocessing stages were time-consuming. DL methods eliminate the necessity of data preprocessing, and they execute well on malware detection issues. With the use of the 1D-CNN, the local feature from network flows was extracted, and LSTM was used to identify the successive relationships among the features. The research of [16] modeled an Android malware detection technique relevant to a hybrid DL method that combined GRU and DBN. Initially, analyzing the Android malware along with deriving static features and dynamic behavioral features having strong anti-obfuscation capability were extracted. Then, construct a hybrid DL technique for detecting Android malware. The work of [17] devised a new TAN (Tree Augmented NB) related hybrid malware detection system utilizing the conditional dependency among related dynamic and static features (system calls, API calls, and permissions), which were needed for the functionality of the application. The author trained three ridge standardized logistic regression techniques in line with the system calls, API calls, and permission of an application and devised their output relation as a TAN (Tree Augmented NB) to identify whether the application was malicious or not.

The study of [18] proposes a hybrid approach for detecting and classifying Android malware, combining static and dynamic analysis methods. It involves three phases: pre-processing, feature selection, and classification. By utilizing features from both types of analysis, the hybrid method enhances accuracy in identifying and categorizing malware compared to traditional approaches. In [19] The authors introduce a six-step framework for identifying IoT malware, addressing challenges in detecting new and modified threats. It incorporates exploratory data analysis, feature engineering, and ensemble learning techniques like GhostNet and Gated Recurrent Unit Ensemble (GNGRUE) trained with the Jaya Algorithm (JA). Extensive testing demonstrates the model's superiority over existing methods, with notable improvements in performance metrics and reduced time complexity. This framework offers cost-effective solutions for detecting various malware strains in IoT environments. The work in [20] addresses the escalating threat of cybercrime and the inefficacy of traditional malware detection methods. It introduces a novel convolutional deep learning neural network designed to accurately detect and classify malware, surpassing existing models in performance.

The approach involves developing a baseline model from scratch, enhancing it through increased depth, and evaluating its effectiveness. The final model achieves a remarkable accuracy rate of 99.183%, demonstrating its superiority in malware detection.

Additionally, the model is tested on new malware samples to validate its efficacy further. Finally, the people of [21] investigated the use of swarm optimization techniques to enhance Android malware detection by identifying key features within API calls. Three optimization methods - Ant Lion Optimization (ALO), Cuckoo Search Optimization (CSO), and Firefly Optimization (FO) - are employed with auto-encoders to identify influential features. These wrapper-based algorithms are evaluated with various machine-learning classifiers. Additionally, a hybrid Artificial Neuronal Classifier (ANC) is proposed to improve classification accuracy. Experimental results demonstrate an impressive 98.87% accuracy using just seven features out of a hundred API call features, indicating significant data optimization.

The literature review of the paper highlights several existing approaches to Android malware detection, each with its own set of limitations:

1. Limited Feature Selection: Many previous studies on Android malware detection rely on a narrow set of feature selections for identifying malware. This limitation restricts the comprehensiveness and effectiveness of the detection process.

2. Resource Constraints for IoT Devices: Traditional malware detection methods often cannot be directly applied to IoT devices due to their limited resources, such as memory, processing capabilities, and computational complexity. This gap presents a challenge in deploying effective malware detection in IoT environments.

3. Dependency on Signature Libraries and Human Analysis: Conventional approaches predominantly depend on **accumulating** signature libraries and require significant human intervention by malware analysts, making it difficult to adapt quickly to the rapidly evolving landscape of Android malware.

4. Preprocessing Complexity: Machine learning (ML) techniques for malware detection typically require extensive data preprocessing, which can be

time-**consuming** and may introduce additional complexity to the detection process.

5. **Vulnerability to Adversarial Attacks**: Some studies have focused on the robustness of Android malware detection systems against adversarial attacks. However, many existing techniques could be compromised, indicating a need for more secure and resilient detection methods.

By addressing these drawbacks, the paper positions the proposed HAFSO-DLMD technique as an advanced solution capable of overcoming the limitations of current approaches through innovative feature selection, optimization for IoT resources, reduced reliance on signature libraries, simplified preprocessing, and enhanced security against adversarial threats.

3. Problem statement

In the rapidly evolving landscape of cloud computing and the Internet of Things (IoT), the Android Operating System (AOS) has become increasingly vulnerable to a wide array of cyberattacks, including financial loss, privacy breaches, unauthorized access, data integrity compromises, and denial of service (DoS). Despite the urgent need for effective cybersecurity measures, existing malware detection solutions for Android devices often struggle to keep pace with the sophistication and volume of new threats, primarily due to reliance on signature libraries and manual analysis. This paper addresses the critical challenge of enhancing malware detection accuracy and efficiency for Android devices within cloud environments. By introducing a novel Hybrid Artificial Fish Swarm Optimization with Deep Learning-Driven Malware Detection (HAFSO-DLMD) technique, we aim to significantly improve the precision of malware identification through deep learning models, thereby strengthening Android's cybersecurity infrastructure against an ever-expanding threat landscape.

4. Materials and methods

In this study, we have focused on the design of a new HAFSO-DLMD technique for malware detection in Android IoT devices. The primary intention of the HAFSO-DLMD technique lies in the proper identification of Android malware using the DL model. The HAFSO-DLMD technique follows a three-stage process: preprocessing, DSAE-based Android malware detection, and HAFSO-based hyperparameter tuning. Fig. 2 showcases the overall flow of the HAFSO-DLMD approach.

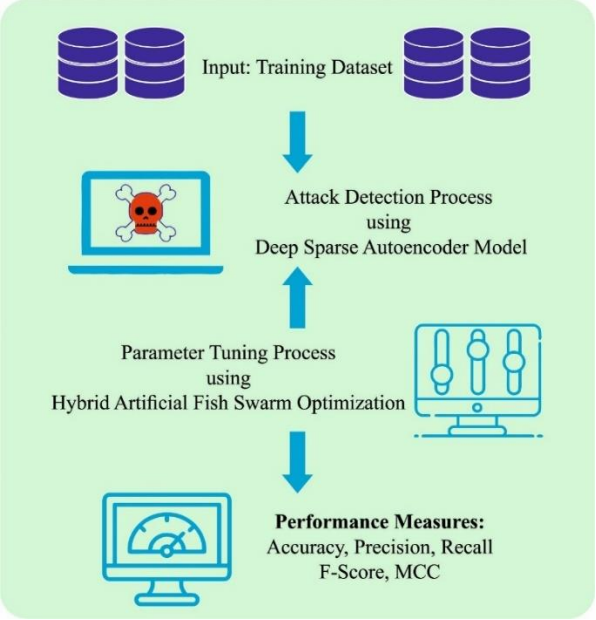


Figure. 2 Overall flow of the HAFSO-DLMD approach

4.1 Preprocessing

As the classes.dex files in dissimilar APKs have different sizes, and the DL model requires a fixed-size input. This input needs to be transformed to a fixed size [19]. Shorter sequences of dissimilar lengths are combined by filling operation. However, longer sequences (particularly above $10MB$) cannot be directly used to input. First, the preprocessing technique reads the classes.dex file of APK as an unknown vector, and later convert those vectors into the fixed size (L) through resampling. The image resampling technique is used for preprocessing the original input; however, they focus primarily on the classification of the malware family. The resampling algorithm generally applied in the image processing field includes Bicubic Interpolation, Nearest Neighbor Interpolation (NNI), and Bilinear Interpolation. Amongst them, NNI employs the gray value of the pixel closer to the sampling point as the target value; hence, the computation becomes simple. Different from 2D images, we apply the same concept for resampling the 1D sequence. Assume that the source vector is $x = \{x_0, \dots, x_n\}$, $n = L_0$, the NNI resampling vector is $A = \{a_0, \dots, a_n\}$, $n = L_1$, then the scaling factor $K = L_0/L_1$ and $a_i = Kx_i$. There is no actual corresponding value in the source vector. However, K cannot be an integer, so K will be rounded to $\lfloor K \rfloor$ to find the values of the nearest neighbor. Meanwhile, the length of the original input vector is generally more prominent than L, viz., the majority of input needs to be down-sampled. The study applied Average-pooling (AVGPOOL) in the downsampling technique for comparison, as well as the upsampling technique still being used in NNI.

4.2 Android malware detection using DSAE model

In the presented HAFSO-DLMD technique, the DSAE model is applied for malware detection purposes. The autoencoder (AE) is a symmetrical network architecture that encompasses input, hidden, and output layers [20]. The AE tries to get the approximate values of the input in the hidden layer (HL) to regenerate the output. The neuron in the first two layers completes the process of encoding, and the last two layers complete the decoding process. Using the backpropagation model, the key features of the input dataset are learned in an unsupervised way by minimalizing the reconstructed error. The encoder and decoder methods are given in the following:

$$H = f_1(W_1X + b_1) \quad (1)$$

$$X' = f_2(W_2H + b_2) \quad (2)$$

From the expression, f_1 and f_2 denote the activation function, X and X' Indicates the input and output units, H shows the hidden unit, W_1 and W_2 represent the weight matrix between the neurons and b_1, $and\ b_2$ indicate the bias of each layer. Even though AE could regenerate the input dataset in the output, the AE doesn't successfully extract the useful feature by copying the input layer to the HLs. The DSAE is an addition of AE, which causes the AE to learn sparse features by presenting the sparsity constraints. In this work, the activation function $f(x) = (1 + e^{-x})^{-1}$, such as the sigmoid activation function, maps the output within the [0,1] interval. Thus, if the output of HL is closer to 0, it is considered an inhibitory state, and the construction of DSAE can be accomplished once the sparsity restriction is imposed on the AE. Mostly, the HL node is in an inhibitory state. Such conditions might enhance the efficiency of the conventional AE and make them better.

Meanwhile, the output regenerates the input, the MSE is exploited for constructing the loss function, as demonstrated in Eq. (3), and the unsupervised training on DSAE can be performed by minimalizing the loss function.

$$I(w, b) = \frac{1}{2}||X' - X||^2 \quad (3)$$

To accomplish the sparsity constraints, the average activation degree of i^{th} neurons in the HL can be represented as

$$\hat{\rho} = \frac{1}{n}\sum_{j=1}^{n} h_i\left(\chi_j\right) \quad (4)$$

From the expression, n signifies the number of neurons in the given problem. For making the activation degree of most neurons in the HL tend to 0, a ρ sparse parameter closer to 0 is presented, with $\rho = \hat{\rho}_i$. In the meantime, a penalty factor is presented based on relative entropy demonstrated in Eq. (5) to discount the case with huge distinctions between $\hat{\rho}_i$ and ρ.

$$KL(\hat{\rho}_i||\rho) = \rho\ log\ \frac{\rho}{\hat{\rho}_i} + (1-\rho) log\ \frac{1-\rho}{1-\hat{\rho}_i} \quad (5)$$

The penalty factor increased with the distinction between $\hat{\rho}_i$ and p, and the value is 0 if $\hat{\rho}_{\mathrm{i}} = \rho$. Then, the overall loss function for the DSAE can be attained by,

$$J_s(W, b) = J(W, b) + \beta \sum_{i=1}^{m} KL\ (\hat{\rho}_i||\rho) \quad (6)$$

In Eq. (6), m characterizes the number of neurons in the HL, and W and b denote the network weight coefficients attained by minimalizing the loss function. And without feature loss, DSAE could decrease the dimension of complex signals.

4.3 Hyperparameter tuning using the HAFSO algorithm

To improve the detection rate of the DSAE model, the HAFSO technique is used to adjust the hyperparameter optimally. Technically, based on the proposed algorithm, the AFSO is composed of 2 essential components, namely parameters and functions connected with the behaviors of fish [21]. The parameter involves the visual distance of individual fish (Visual), the crowd factor of fish (δ), the size of fish movement (Step), and the distance between the two fish signify the $\mathrm{X_i}$ & $\mathrm{X_j}(\mathrm{d_{ij}} = \|\mathrm{X_i} - \mathrm{X_j}\|)$, where $X = (X_1, X_2, \ldots, X_n)$ & $Y = f(x)$. X represents an individual condition in the fish population, and Y specifies objective functions or feeds focus value. Step and Visual play a key role among the four parameters. The larger the value of the parameter, the faster AFSO moves toward the global optimum; meanwhile, fish moves large steps at every iteration and examines the largest space near them. Search, Follow, and Swarm are the three behaviors of fish that are transformed into essential functions in AFSO. Once the fish finds the region using a high food concentration, it directly goes towards that region.

When the fish discovers a lot of food, then others will share it. During AFSO adaption, a higher concentration of feed is compared to the current

situation. When there is another situation, then the fish moves toward it and assures the existence of a colony. They tend to swarm, apparently, to prevent the risks. The Step and Visual values have a massive effect on the fish's behaviors. For example, when the size of the Visual is narrow, then searching and swarming behavior can be controlled. In the Visual, The fish tends to find high feed focus. Then, the swarming performance would be determined by whether the fish moves towards it or not. The higher the value of Step and Visual, the faster global optima was obtained. This concept inspired researchers to employ AFSO to resolve different optimization issues. The mathematical expression of fish swarming is shown below:

$$X = (X_1, X_2, X_3, \ldots, X_n) \tag{7}$$

In Eq. (7), X signifies the fish, and the visual place is represented as follows:

$$X_v = (X_{v1}, X_{v2}, X_{v3}, \ldots, X_{vn)} \tag{8}$$

In Eq. (8), X_v signifies the fish's location in Visual. The abovementioned tasks (7) has performed as follows:

$$X_{vi} = X_{vi} + Rand(\,) \times step \;\text{ if } f(X_{vi}) > f(X_v) \tag{9}$$

$i \in 1, 2, .., n$ In Eq. (9), X_{vi} denotes the fish condition in Visual.

$$X_{next} = X + \left[\frac{(X_v - X)}{(\|X_v - X\|)}\right] \times Step \times Rand() \tag{10}$$

In Eq. (10), X_{next} signifies the subsequent fish in Visual. To the abovementioned formula, Eq. (7) shows the state of the fish, Eq. (8) represents the fish location in Visual, and Eq. (9) specifies how Eqs. (7) and (8) work together, and it demonstrates the cowardly factors in AFSO. Eq. (10) shows the subsequent fish is determined by the distance between the value of Step and the two fishes.

In Eq. (9), the Search Function represents the search behaviors. When $X_i < X_j$ then (12) is implemented. Next, X_i and X_j indicate the existing and subsequent food concentrations.

$$X_i^{(t+i)} = X_i^t + \frac{X_i - X_i^{(t)}}{\left(\left\|X_i - X_i^{(t)}\right\|\right)} \times Step \times Rand(\,) \tag{11}$$

Or else Carry out (9) by arbitrarily choosing the state X_i and inspect the outcomes using Eq. (11). Once it does not meet even after try_number, t ($t <$ Search Function), then it arbitrarily moves a step, which makes them escape from the local extrema as,

$$X_i^{(t+i)} = X_i^t + Visual \times Rand() \tag{12}$$

$$X_i^{(t+i)} = X_i^t + +[\frac{X_j - X_i^{(t)}}{\left(\left\|X_c - X_c^{(t)}\right\|\right)} \times Step \times Rand(\,) \tag{13}$$

whereas Eq. (14) is called the $Swarm\ Function$ where it represents swarming behaviours. Carry out (13) once it satisfies all the conditions. The current state of the point is X_i ($d_{ij} <$ Visual), or else perform the Search Function:

i. $(\mathrm{n_f}/\mathrm{n}) < \delta$
ii. $X_c > X_i$; X_c signifies the central food concentration

$$X_i^{t+1} = X_i^t + \left[\frac{\left(X_j - X_i^t\right)}{\left(\left\|X_j - X_i^t\right\|\right)}\right] \times Step \times Rand(); \tag{14}$$

whereas $Follow\ Function$ represents the following behavior. Carry out (14) if the whole conditions are met, or else perform Search Function: The current state of the point is $X_i(d_{ij} <$*Visual*$)$, $(\mathrm{n_f}/\mathrm{n}) < \delta$ and $X_j > X_i$. Swarm, Follow, and Search behaviors are correspondingly symbolized in Eqs. (12)-(14). They would be carried out afterward, and the related state would be fulfilled. Then Eqs. (11) and (12) are implemented. The process is continued until the condition is met by repeating Eq. (9). The process is repeated till the optimal point is obtained. If the criteria are met, then the existing optima value based on obtained outcomes would be upgraded. At last, if the ending condition is met, the outcome is recorded.

The use of a chaotic concept designs the HAFSO algorithm. In practice, the chaotic logistic system has complex dynamic behavior and is more commonly used. The chaotic system has the properties of being sensitive to the initial condition. The disorganized signal produced by the deterministic system has the quality of genus-randomness, and the initial value and chaos mapping parameter can define the curve.

$$\lambda_{i+1} = \mu \times \lambda_i \times (1 - \lambda_i) \tag{15}$$

where $\lambda \in [0,1], i = 0,1,2, \cdots, \mu$ is within [1, 4]. The study recommended that μ is closest to 4, $and\ \lambda$ is most relative to the average distribution within [0,1].

In contrast, the system is chaotic if the μ value is 4. The initial population plays a crucial role in intelligent optimization techniques that affect the final solution quality and the convergence rate. Here, Logistic chaotic mapping initializes the population of the AFSO technique, which exploits solution space to improve the efficiency of the algorithm.

The fitness selection is a critical factor in the HAFSO algorithm. Solution encoding is used to assess the aptitude (goodness) of the candidate solution. Here, the accuracy value is the primary condition applied for proposing a fitness function.

$$Fitness = max(P) \quad (16)$$

$$P = \frac{TP}{TP+FP} \quad (17)$$

From the expression, TP represents the true positive, and FP denotes the false positive value.

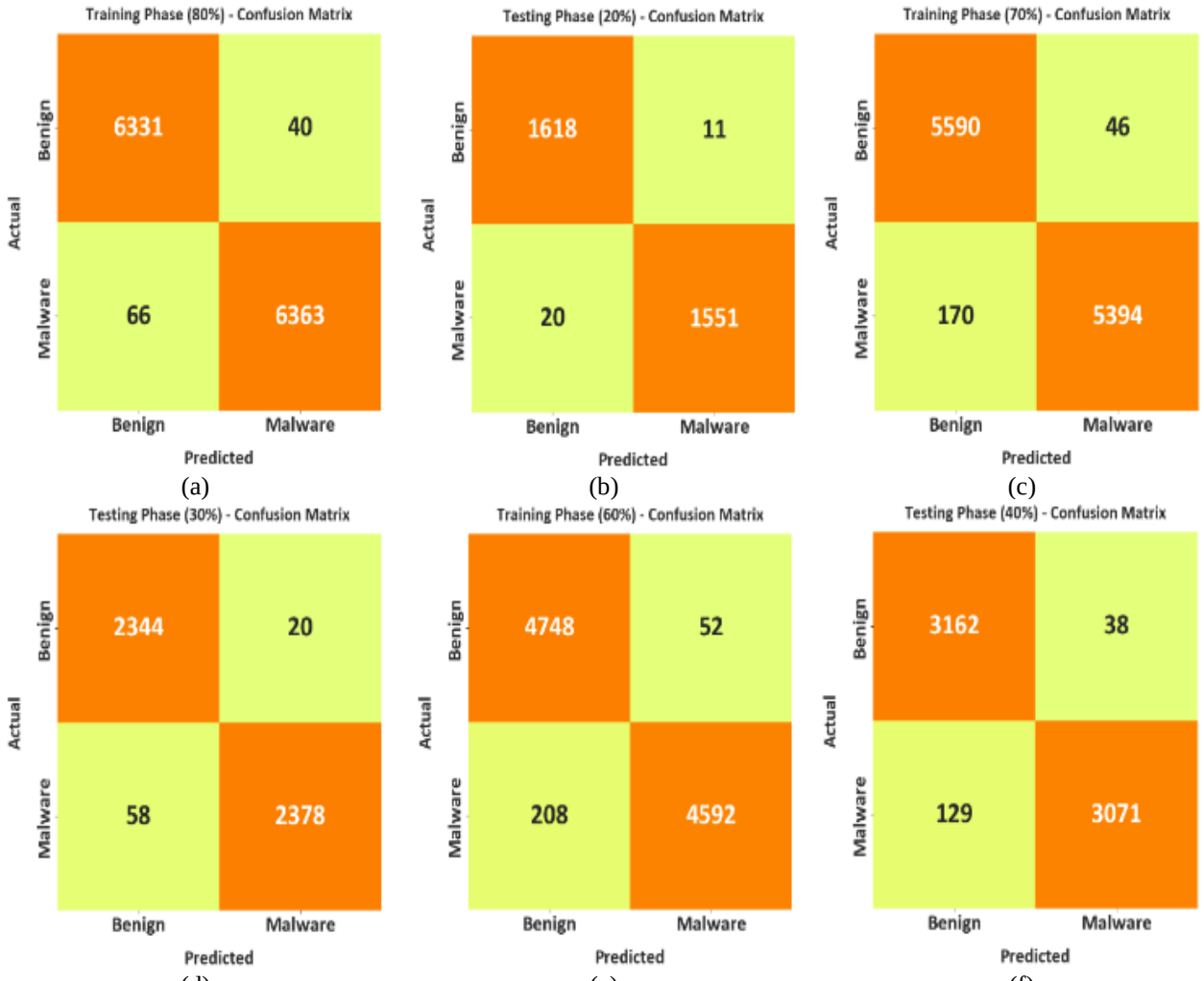
5. Results and discussion

In this section, the results are assessed on the Android malware dataset, comprising 16000 samples with two classes as determined in Table 1.

The confusion matrices of the HAFSO-DLMD method can be examined in Fig. 3. The outcomes indicate that the HAFSO-DLMD system identified the benign and malware samples accurately. For instance, with 80% of the training set (TRS),

Table 1. Details of the dataset

Class	No. of Instances
Benign	8000
Malware	8000
Total Number of Instances	**16000**



(a) (b) (c)

(d) (e) (f)

Figure. 3 Confusion matrices of TRS/TSS of 80:20, TRS/TSS of 70:30, and TRS/TSS of 60:40: (a)Training (80%)-Confusion Matrix, (b)Testing Phase (20%)-Confusion Matrix, (c)Training Phase (70%)-Confusion Matrix, (d)Testing Phase (30%)-Confusion Matrix, (e) Training Phase (60%)-Confusion Matrix, and (f) Testing Phase (40%)-Confusion Matrix

Table 2. Classifier outcome of HAFSO-DLMD approach on 80:20 of TRS/TSS

Class	$Accu_{bal}$	$Prec_n$	$Reca_l$	F_{score}	MCC
Training Phase (80%)					
Benign	99.37	98.97	99.37	99.17	98.34
Malware	98.97	99.38	98.97	99.17	98.34
Average	**99.17**	**99.17**	**99.17**	**99.17**	**98.34**
Testing Phase (20%)					
Benign	99.32	98.78	99.32	99.05	98.06
Malware	98.73	99.30	98.73	99.01	98.06
Average	**99.03**	**99.04**	**99.03**	**99.03**	**98.06**

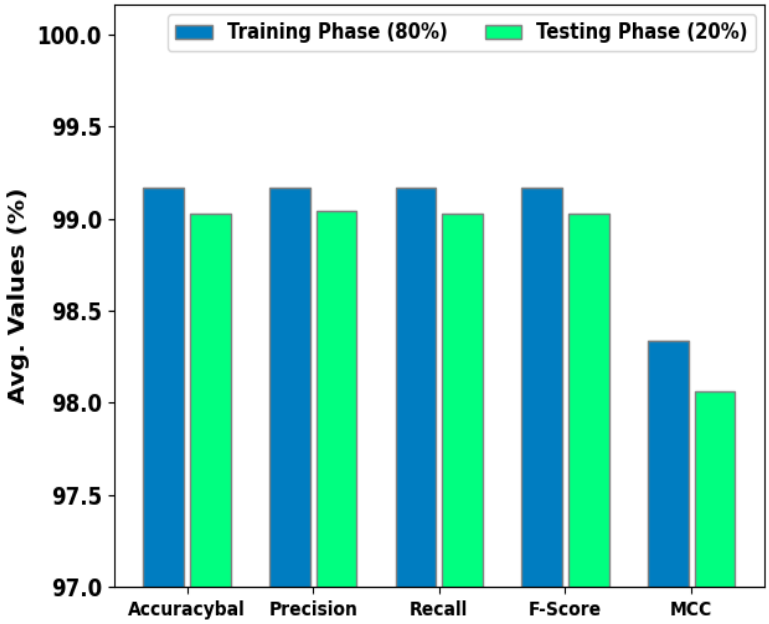


Figure. 4 Average outcome of HAFSO-DLMD approach on 80:20 of TRS/TSS

Table 3. Classifier outcome of HAFSO-DLMD approach on 70:30 of TRS/TSS

Class	$Accu_{bal}$	$Prec_n$	$Reca_l$	F_{score}	MCC
Training Phase (70%)					
Benign	99.18	97.05	99.18	98.10	96.17
Malware	96.94	99.15	96.94	98.04	96.17
Average	**98.06**	**98.10**	**98.06**	**98.07**	**96.17**
Testing Phase (30%)					
Benign	99.15	97.59	99.15	98.36	96.76
Malware	97.62	99.17	97.62	98.39	96.76
Average	**98.39**	**98.38**	**98.39**	**98.37**	**96.76**

the HAFSO-DLMD technique recognizes 6,331 benign samples and 6,363 malware samples. Simultaneously, with 20% of the testing set (TSS), the HAFSO-DLMD method identifies 1618 benign samples and 1551 malware samples. Also, with 70% of TRS, the HAFSO-DLMD system recognizes 5,590 benign samples and 5,394 malware samples. At last, with 60% of TRS, the HAFSO-DLMD approach recognizes 4748 benign samples and 4592 malware samples. In Table 2 and Fig. 4, the overall classifier outcomes of the HAFSO-DLMD method can be examined under 80:20 of TRS/TSS. The results indicate that the HAFSO-DLMD technique identifies the benign and malware samples. For instance, with 80% of TRS, the HAFSO-DLMD technique achieves an average $accu_{bal}$ of 99.17%, $prec_n$ of 99.17%, $reca_l$ of 99.17%, F_{score} of 99.17%, and MCC of 98.34%. Meanwhile, with 20% of TSS, the HAFSO-

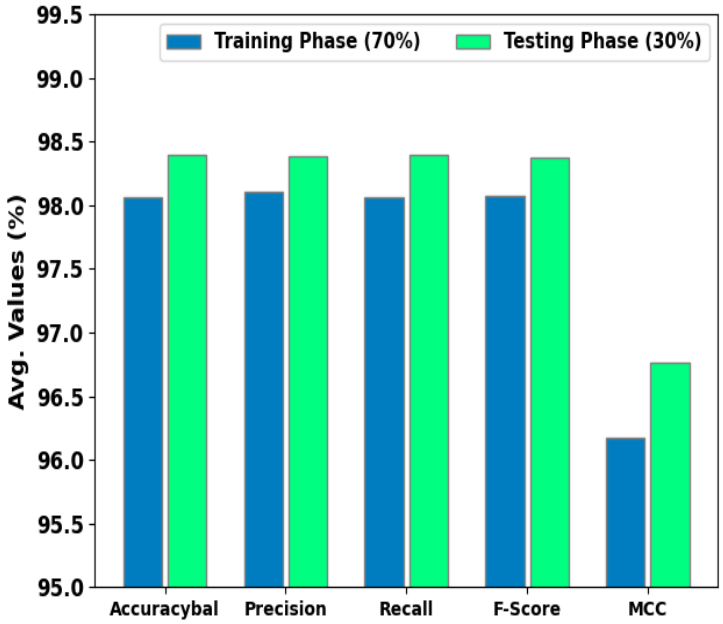


Figure. 5 Classifier outcome of HAFSO-DLMD approach on 70:30 of TRS/TSS

Table 4. Classifier outcome of HAFSO-DLMD approach on 60:40 of TRS/TSS

Class	$Accu_{bal}$	$Prec_n$	$Reca_l$	F_{score}	**MCC**
Training Phase (60%)					
Benign	98.92	95.80	98.92	97.33	94.63
Malware	95.67	98.88	95.67	97.25	94.63
Average	**97.29**	**97.34**	**97.29**	**97.29**	**94.63**
Testing Phase (40%)					
Benign	98.81	96.08	98.81	97.43	94.82
Malware	95.97	98.78	95.97	97.35	94.82
Average	**97.39**	**97.43**	**97.39**	**97.39**	**94.82**

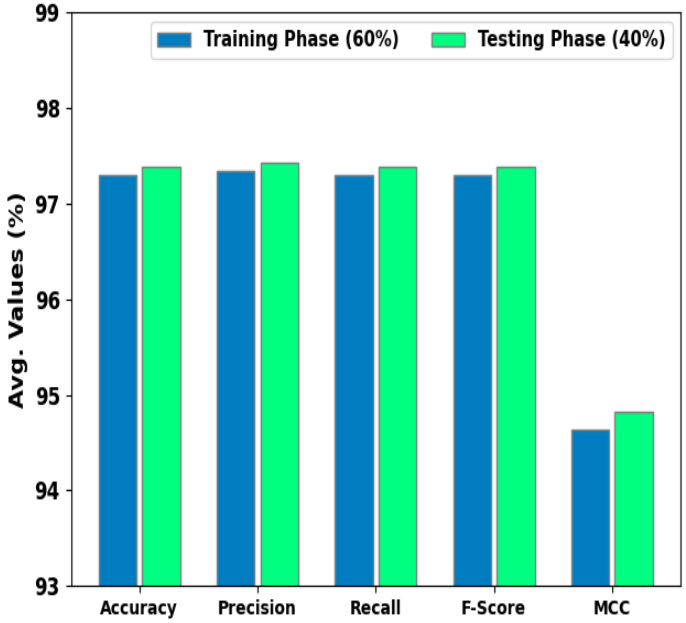


Figure. 6 Classifier outcome of HAFSO-DLMD approach on 80:20 of TRS/TSS

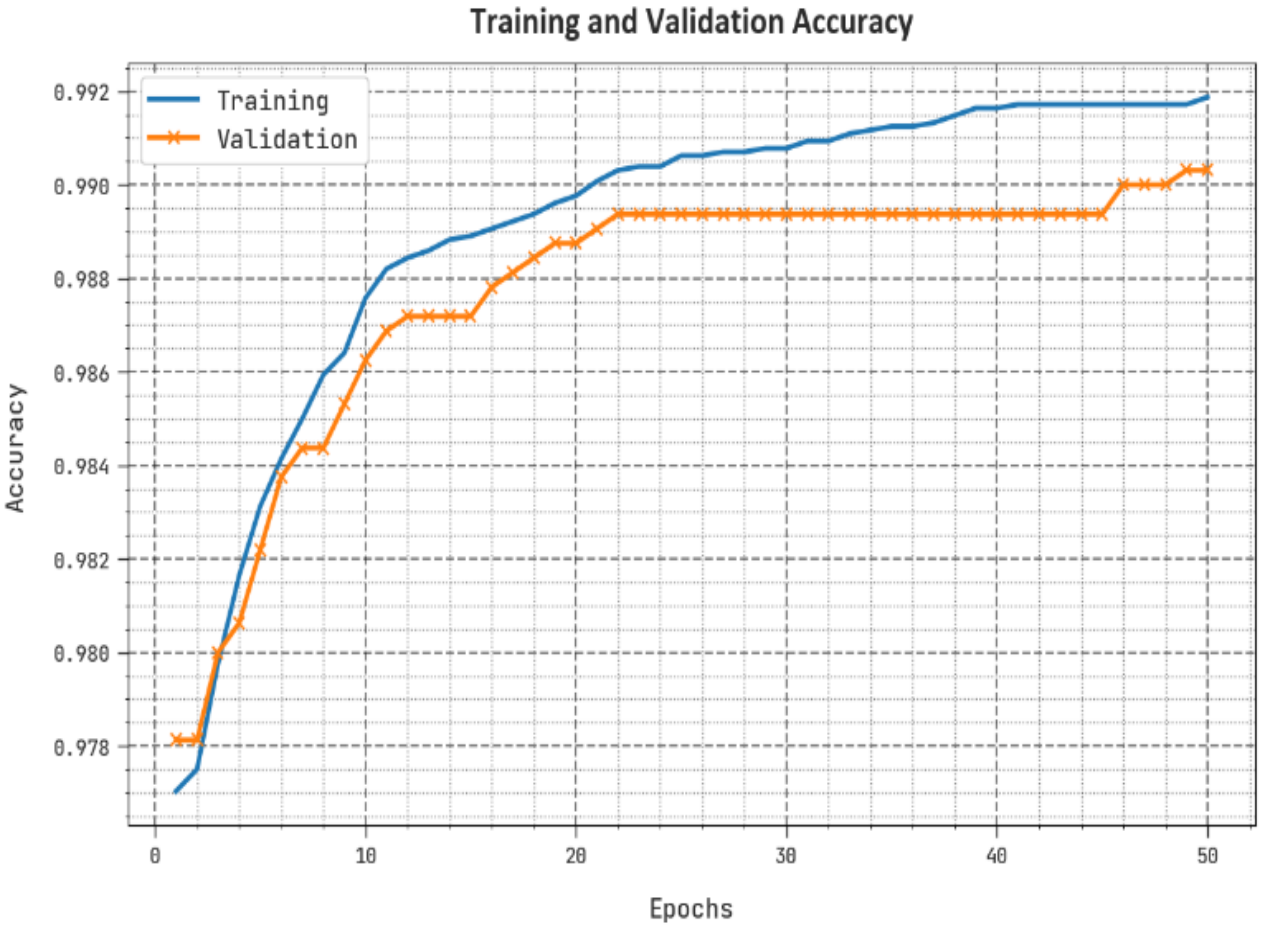


Figure. 7 TACY and VACY outcome of the HAFSO-DLMD approach

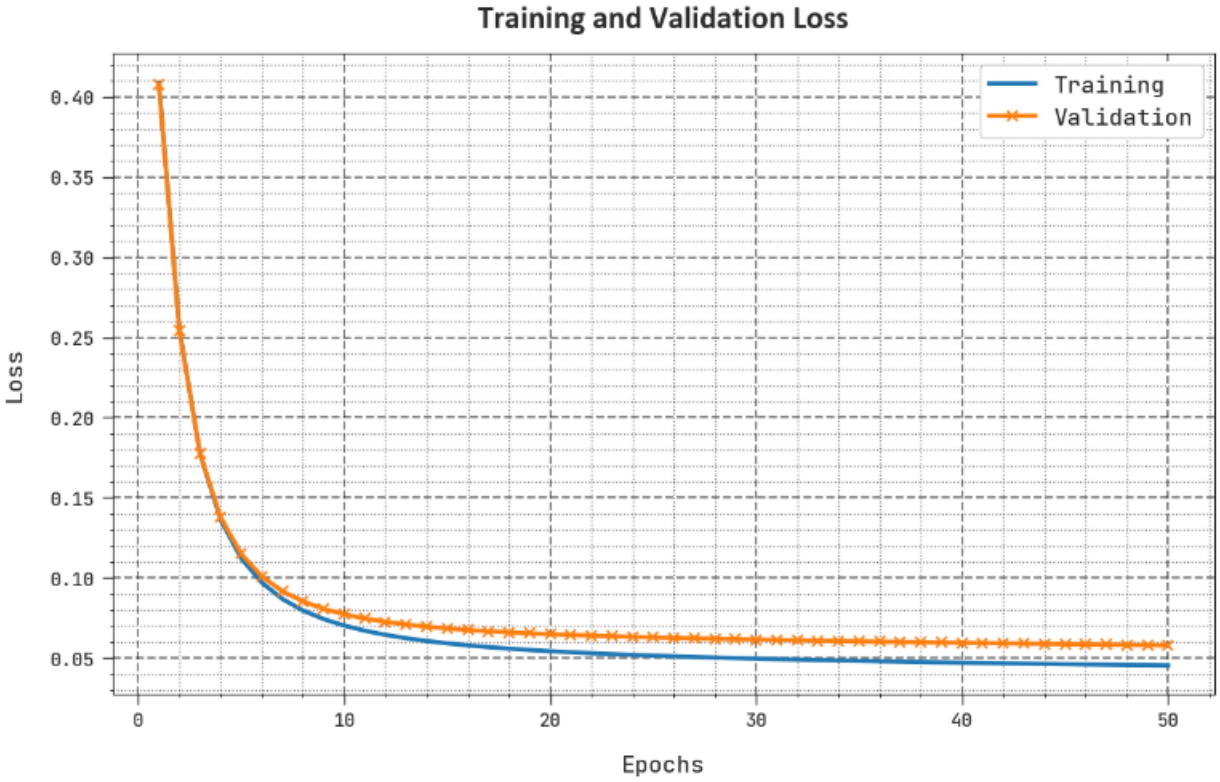


Figure. 8 TLOS and VLOS outcome of the HAFSO-DLMD approach

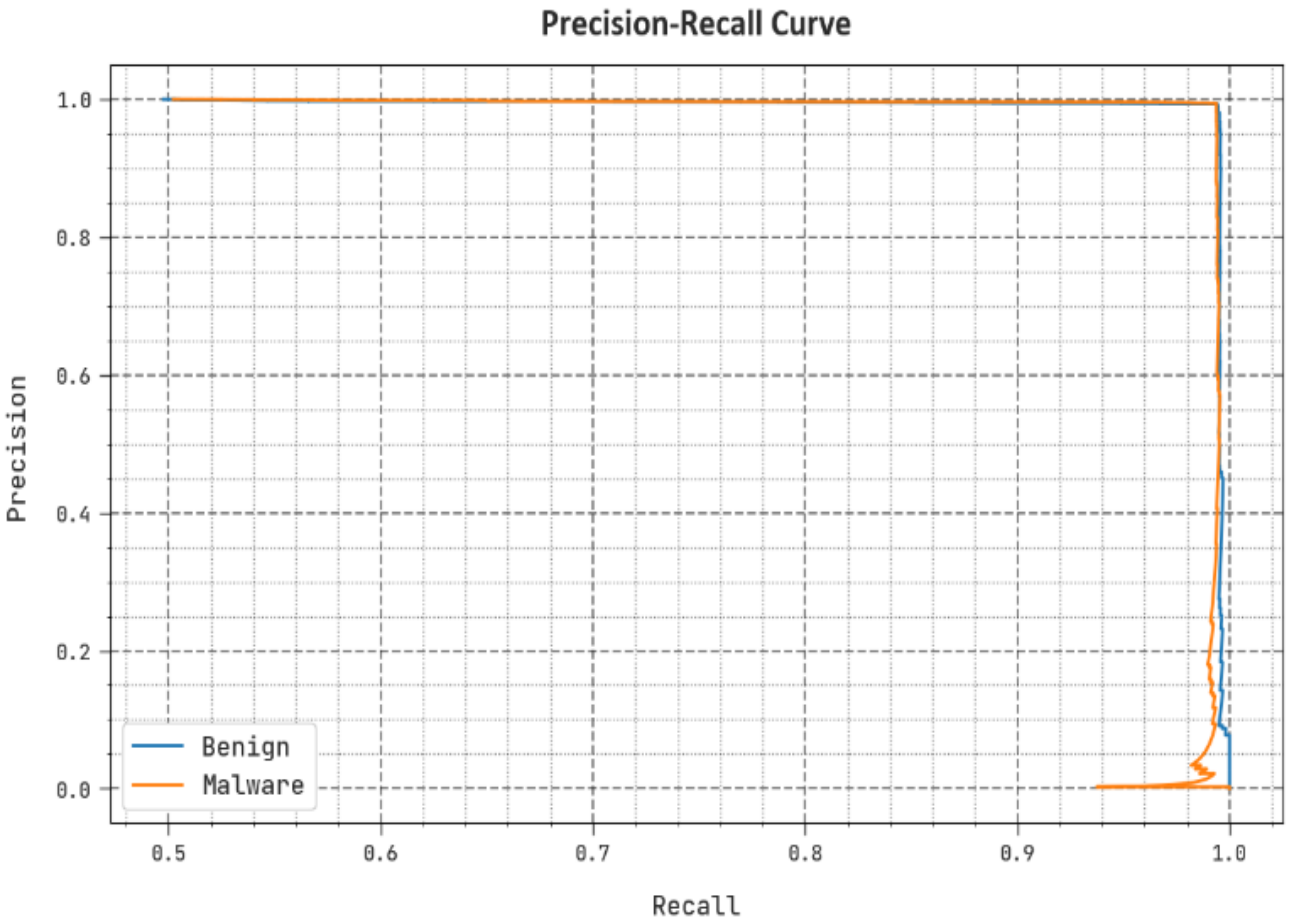


Figure. 9 Precision-recall outcome of the HAFSO-DLMD approach

DLMD method attains an average $accu_{bal}$ of 99.03%, $prec_n$ of 99.04%, $reca_l$ of 99.03%, F_{score} of 99.03%, and MCC of 98.06%.

In Table 3 and Fig. 5, the overall classifier outcomes of the HAFSO-DLMD technique can be examined under 70:30 of TRS/TSS. The outcomes indicate that the HAFSO-DLMD technique identifies the benign and malware samples. For example, with 70% of TRS, the HAFSO-DLMD method achieves an average $accu_{bal}$ of 98.06%, $prec_n$ of 98.10%, $reca_l$ of 98.06%, F_{score} of 98.07%, and MCC of 96.17%. In the meantime, with 30% of TSS, the HAFSO-DLMD approach gains an average $accu_{bal}$ of 98.39%, $prec_n$ of 98.38%, $reca_l$ of 98.39%, F_{score} of 98.37%, and MCC of 96.76%.

In Fig. 6 and Table 4, the overall classifier outcomes of the HAFSO-DLMD method can be examined under 60:40 of TRS/TSS. The results indicate that the HAFSO-DLMD technique identifies the benign and malware samples. For example, with 70% of TRS, the HAFSO-DLMD method achieves an average $accu_{bal}$ of 97.29%, $prec_n$ of 97.34%, $reca_l$ of 97.29%, F_{score} of 97.29%, and MCC of 94.63%. In the meantime, with 30% of TSS, the HAFSO-DLMD technique achieves an average $accu_{bal}$ of 97.39%, $prec_n$ of 97.43%, $reca_l$ of 97.39%, F_{score} of 97.39%, and MCC of 94.82%.

The TACY and VACY of the HAFSO-DLMD technique are investigated on malware detection performance in Fig. 7. The figure implied that the HAFSO-DLMD approach has demonstrated superior performance with increased values of TACY and VACY. Notably, the HAFSO-DLMD model has the highest TACY outcomes. The TLOS and VLOS of the HAFSO-DLMD technique are tested on malware detection performance in Fig. 8. The figure inferred that the HAFSO-DLMD method has revealed superior performance with the least values of TLOS and VLOS. Visibly, the HAFSO-DLMD approach has minimum VLOS outcomes.

A precise precision-recall examination of the HAFSO-DLMD technique under the test database is illustrated in Fig. 9. The figure shows the HAFSO-DLMD algorithm has superior precision-recall values under all classes.

Finally, enhanced performance of the HAFSO-DLMD technique can be ensured by a comparison study in Table 5 with other works [22]. The experimental values highlighted that the HAFSO-DLMD technique reaches effectual outcomes with

Table 5. Comparative analysis of the HAFSO-DLMD approach with other systems [22]

Methods	$Accu_{bal}$	$Prec_n$	$Reca_l$	F_{score}
HAFSO-DLMD	99.17	99.17	99.17	99.17
DexCNN	93.75	90.12	98.61	93.95
DexCRNN_GRU	95.90	95.56	95.76	95.89
DexCRNN_LSTM	94.16	92.46	96.18	94.29
DexCRNN_BiGRU	96.18	95.63	96.56	96.12
DexCRNN_BiLSTM	95.45	94.61	96.31	95.36

Table 6. Symbols definitions.

Symbol	Meaning
f_1 and f_2	Activation functions
n	Number of neurons
ρ	Sparse parameter
KL	Penalty factor
X_v	Fish's location
X_{next}	The subsequent fish in Visual
X_i	The existing food concentrations
X_j	The subsequent food concentrations
X_c	The central food concentration
$J_s(W,b)$	The loss function
X_{next}	The subsequent fish in Visual
TSS	Testing set
TRS	Training set

maximum performance. It is noticed that the HAFSO-DLMD technique accomplishes a higher $accu_y$ of 99.17%, $prec_n$ of 99.17%, $reca_l$ of 99.17%, and $F1_{score}$ of 99.17%. These results and discussion guaranteed that the HAFSO-DLMD technique results in maximum performance over other models.

The HAFSO-DLMD method demonstrated superior performance across all evaluated metrics, achieving an accuracy, precision, recall, and F-score of 99.17% in our testing phase. This is a significant improvement over the methods compared, which underscores the effectiveness of integrating Hybrid Artificial Fish Swarm Optimization with Deep Learning for malware detection. The specifics of these comparisons are detailed in Section 5 of our paper, including a comprehensive discussion of the factors contributing to the observed performance improvements. Our method's novelty lies in the unique integration of the HAFSO algorithm for optimal hyperparameter tuning of the Deep Sparse Autoencoder (DSAE) model, which is pivotal in processing the complex patterns inherent in Android malware. This synergy between advanced optimization techniques and deep learning significantly enhances the detection rate, setting a new benchmark for malware detection in Android devices. The proposed HAFSO-DLMD technique addresses specific challenges that were not fully resolved by existing approaches, including the dynamic nature of malware evolution and the complex landscape of Android applications. Our method's adaptability and precision in detecting sophisticated malware strains represent a considerable step forward in cybersecurity efforts. Table 6 list the meaning of the symbols used throughout the work.

6. Conclusion

In this study, we have introduced the HAFSO-DLMD technique for efficient malware detection in Android devices within cloud environments. The HAFSO-DLMD technique, which harnesses a deep learning model, has been meticulously designed to accurately identify Android malware. This process began with preprocessing the raw bytecodes from the classes.dex file of Android applications. We applied the DSAE model for the actual detection of malware, with the HAFSO algorithm fine-tuning the hyperparameters to achieve optimal performance. Our rigorous experimental evaluation, conducted on an Android APK dataset, demonstrated that the HAFSO-DLMD technique achieved impressive accuracy, precision, recall, and F_{score} of 99.17% in the training phase (80% of the dataset) and accuracy, precision, recall, and F_{score} of 99.03% in the testing phase (20% of the dataset), outperforming other recent approaches. These results solidify the

HAFSO-DLMD method as a significant advancement in the domain of Android cybersecurity. Future work could explore the development of ensemble deep learning-based malware detection techniques to further enhance the detection rate.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, A.A., I.K.I., A.T.A., M.A.A., I.A.H.; Methodology, A.A., I.K.I., A.T.A., M.A.A., I.A.H; Software, A.A., I.K.I., A.T.A., M.A.A., I.A.H; Validation, A.A., I.K.I., A.T.A., M.A.A., I.A.H; Formal analysis, A.T.A., M.A.A., I.A.H; Investigation, A.A., I.K.I.,; Resources, A.A., I.K.I., A.T.A., M.A.A., I.A.H; Writing—original draft, A.A., I.A.H; Writing—review & editing, A.A., I.K.I., A.T.A., M.A.A., I.A.H.; Visualization, A.A., I.K.I., A.T.A., M.A.A., I.A.H.; Supervision, I.K.I. All authors have read and agreed to the published version of the manuscript.

Acknowledgments

The authors would like to acknowledge the support of Prince Sultan University for paying the Article Processing Charges (APC) of this publication. Special acknowledgment to Automated Systems and Soft Computing Lab (ASSCL), Prince Sultan University, Riyadh, Saudi Arabia. In addition, the authors wish to acknowledge the editor and anonymous reviewers for their insightful comments, which have improved the quality of this publication.

References

[1] M. A. Omer, S. R. Zeebaree, M. A. Sadeeq, B. W. Salim, S. X. Mohsin, Z. N. Rashid, and L. M. Haji, "Efficiency of malware detection in android system: A survey", *Asian Journal of Research in Computer Science*, Vol. 7, No. 4, pp.59-69, 2021.

[2] J. Abawajy, A. Darem, and A. A. Alhashmi, "Feature subset selection for malware detection in smart IoT platforms", *Sensor*, Vol. 21, No. 4, p. 1374, 2021.

[3] G. D'Angelo, F. Palmieri, A. Robustelli, and A. Castiglione, "Effective classification of android malware families through dynamic features and neural networks", *Connection Science*, Vol. 33, No. 3, pp. 786-801, 2021.

[4] J. Jung, J. Park, S.J. Cho, S. Han, M. Park, and H.H. Cho, "Feature engineering and evaluation for android malware detection scheme", *Journal of Internet Technology*, Vol. 22, No. 2, pp. 423-440, 2021.

[5] A. Mahindru and A.L. Sangal, "MLDroid—Framework for Android malware detection using machine learning techniques", *Neural Computing and Applications*, Vol. 33, No. 10, pp. 5183-5240, 2021.

[6] 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.

[7] H. Zhu, Y. Li, R. Li, J. Li, Z. You, and H. Song, "SEDMDroid: An enhanced stacking ensemble framework for Android malware detection", *IEEE Transactions on Network Science and Engineering*, Vol. 8, No. 2, pp. 984-994, 2020.

[8] Q. D. Ngo, H. T. Nguyen, V. H. Le, and D. H. Nguyen, "A survey of IoT malware and detection methods based on static features", *ICT Express*, Vol. 6, No. 4, pp. 280-286, 2020.

[9] S. Baek, J. Jeon, B. Jeong, and Y. S. Jeong, "Two-stage hybrid malware detection using deep learning", *Human-centric Computing and Information Sciences*, Vol. 11, No. 27, 2021.

[10] R. Taheri, M. Shojafar, M. Alazab, and R. Tafazolli, "FED-IIoT: A robust federated malware detection architecture in industrial IoT", *IEEE Transactions on Industrial Informatics*, Vol. 17, No. 12, pp. 8442-8452, 2020.

[11] G. Dhabal and G. Gupta, "Towards Design of a Novel Android Malware Detection Framework Using Hybrid Deep Learning Techniques", *in Soft Computing for Security Applications*, Springer, Singapore, pp. 181-193, 2023.

[12] J. Kim, Y. Ban, E. Ko, H. Cho, and J. H. Yi, "MAPAS: a practical deep learning-based android malware detection system", *International Journal of Information Security*, pp. 725–738, 2022, Doi:10.1007/s10207-022-00579-6.

[13] S. Fallah and A. J. Bidgoly, "Android malware detection using network traffic based on sequential deep learning models", *Software: Practice and Experience*, Vol. 52, No. 9, pp. 1987-2004, 2022.

[14] H. Rathore, S.K. Sahay, P. Nikam, and M. Sewak, "Robust android malware detection system against adversarial attacks using q-learning", *Information Systems Frontiers*, Vol. 23, No. 4, pp. 867-882, 2021.

[15] M. Gohari, S. Hashemi, and L. Abdi, "Android malware detection and classification based on network traffic using deep learning", In: *Proc. of the 2021 7th International Conference on Web*

Research (ICWR), Tehran, Iran, pp. 71-77, 2021, doi: 10.1109/ICWR51868.2021.9443025.

[16] T. Lu, Y. Du, L. Ouyang, Q. Chen, and X. Wang, "Android Malware Detection Based on a Hybrid Deep Learning Model", *Security and Communication Networks*, Vol. 2020, Article ID 8863617, 2020, doi: 10.1155/2020/8863617.

[17] R. Surendran, T. Thomas, and S. Emmanuel, "A TAN based hybrid model for android malware detection", *Journal of Information Security and Applications*, Vol. 54, p. 102483, 2020.

[18] F. Taher, O. AlFandi, M. Al-kfairy, H. Al Hamadi, and S. Alrabaee, "DroidDetectMW: A Hybrid Intelligent Model for Android Malware Detection", *Applied Sciences*, Vol. 13, No. 13, p. 7720, 2023, doi: 10.3390/app13137720.

[19] A. A. Almazroi and N. Ayub, "Enhancing Smart IoT Malware Detection: A GhostNet-based Hybrid Approach", *Systems*, Vol. 11, No. 11, p. 547, 2023, doi: 10.3390/systems11110547.

[20] M. Omar, "New Approach to Malware Detection Using Optimized Convolutional Neural Network", *in *Machine Learning for Cybersecurity**, SpringerBriefs in Computer Science. Springer, Cham, 2022, doi: 10.1007/978-3-031-15893-3_2.

[21] K. S. Jhansi, P. R. K. Varma, and S. Chakravarty, "Swarm Optimization and Machine Learning for Android Malware Detection", *Comput. Mater. Contin.*, Vol. 73, No. 3, pp. 6327-6345, 2022, doi: 10.32604/cmc.2022.030878.

[22] Z. Ren, H. Wu, Q. Ning, I. Hussain, and B. Chen, "End-to-end malware detection for Android IoT devices using deep learning", *Ad Hoc Networks*, Vol. 101, pp. 102098, 2020.