Intelligent Automation & Soft Computing
DOI: 10.32604/iasc.2023.034907
Article

Tech Science Press

Graph Convolutional Neural Network Based Malware Detection in IoT-Cloud Environment

Faisal S. Alsubaei[1], Haya Mesfer Alshahrani[2], Khaled Tarmissi[3] and Abdelwahed Motwakel[4,*]

[1]Department of Cybersecurity, College of Computer Science and Engineering, University of Jeddah, Jeddah, 21959, Saudi Arabia
[2]Department of Information Systems, College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University, P.O.Box 84428, Riyadh, 11671, Saudi Arabia
[3]Department of Computer Sciences, College of Computing and Information System, Umm Al-Qura University, Makkah, 24211, Saudi Arabia
[4]Department of Computer and Self Development, Preparatory Year Deanship, Prince Sattam bin Abdulaziz University, AlKharj, 16242, Saudi Arabia
*Corresponding Author: Abdelwahed Motwakel. Email: a.ismaeil@psau.edu.sa
Received: 31 July 2022; Accepted: 26 October 2022

Abstract: Cybersecurity has become the most significant research area in the domain of the Internet of Things (IoT) owing to the ever-increasing number of cyberattacks. The rapid penetration of Android platforms in mobile devices has made the detection of malware attacks a challenging process. Furthermore, Android malware is increasing on a daily basis. So, precise malware detection analytical techniques need a large number of hardware resources that are significantly resource-limited for mobile devices. In this research article, an optimal Graph Convolutional Neural Network-based Malware Detection and classification (OGCNN-MDC) model is introduced for an IoT-cloud environment. The proposed OGCNN-MDC model aims to recognize and categorize malware occurrences in IoT-enabled cloud platforms. The presented OGCNN-MDC model has three stages in total, such as data pre-processing, malware detection and parameter tuning. To detect and classify the malware, the GCNN model is exploited in this work. In order to enhance the overall efficiency of the GCNN model, the Group Mean-based Optimizer (GMBO) algorithm is utilized to appropriately adjust the GCNN parameters, and this phenomenon shows the novelty of the current study. A widespread experimental analysis was conducted to establish the superiority of the proposed OGCNN-MDC model. A comprehensive comparison study was conducted, and the outcomes highlighted the supreme performance of the proposed OGCNN-MDC model over other recent approaches.

Keywords: Cybersecurity; IoT; cloud; malware detection; graph convolution network

1 Introduction

The Internet of Things (IoT) is a novel network paradigm in which devices, machines and human beings communicate and collaborate with each other through novel association procedures [1]. The pervasive and

ever-rising cybersecurity assaults on IoT mechanisms made organizations and people experience a wide range of complexities in finance, reputation, compliances and day-to-day business functions [2]. The increased occurrences of cyberattacks are a direct result of phenomenal growth and the development of IoT gadgets in these fields in terms of smart manufacturing, smart grids, patient monitoring systems, logistics and environmental monitoring. It is challenging to accomplish the security management of IoT networks due to the transient and dynamic nature of the links among the devices, the diversity of the players who can interact with IoT networks and resource limitations [3]. The global IoT security market is anticipated to expand at an Annual Growth Rate of 33.7%, owing to multiple factors such as a high number of cyberattacks on IoT gadgets, heavy regulations on IoT security and an increased number of security concerns [4].

In recent years, mobile devices have turned out to be a crucial part of everyone's life and are utilized even more than traditional computers like personal computers [5]. The data generated by smartphones is different and highly confidential in nature. Thus, smartphones are employed as working tools, payment means and also for other modes of interaction [6]. On the other hand, these devices are highly susceptible to cyberattacks than conventional computer systems since all the types of protocols and networks are used in these devices, such as mobile networks, Wi-Fi (e.g., 3rd or 4th generation), etc. As mentioned earlier, the rapid penetration of Android platforms in mobile devices makes detecting malware assaults a significant task to accomplish [7]. This malware can access the data of the users from their Android devices, transmit Short Message Services (SMSs) to the usernames and disrupt the privacy of a user [8]. Such evolved malware have been contained earlier using numerous analytical techniques and malware identification processes like hybrid analysis, static analysis and dynamic analysis to secure Android devices from cyberattacks [9]. Thus, a precise malware detection technique needs a large volume of hardware resources that are not only highly significant but also must be resource-limited for mobile devices [10]. This denotes the requirement for a multi-objective malware detection technique that can function in mobile atmospheres and optimally solve this problem.

In this article, an optimal Graph Convolutional Neural Network-based Malware Detection and Classification (OGCNN-MDC) model is introduced for the IoT-cloud environment. The proposed OGCNN-MDC aims to recognize and categorize the occurrences of malware in IoT-enabled cloud platforms. The presented OGCNN-MDC model has three stages in total, such as data pre-processing, malware detection and parameter tuning. To detect and classify the malware, the GCNN model is exploited in this study. To enhance the overall efficiency of the GCNN model, the Group Mean-Based Optimizer (GMBO) algorithm is utilized for appropriate adjustment of the GCNN parameters. A widespread experimental analysis was conducted to highlight the improvements of the proposed OGCNN-MDC model.

2 Related Works

In the literature [11], the authors developed an event-aware and scalable Android malware detection technique called EveDroid. This technique exploited the behavioural patterns of different activities to efficiently identify the novel malware in accordance with the insights reflected by the events in case of potential malware attacks [11]. Unlike the existing methodologies that use Application Programming Interface (API) calls as features directly, the authors developed this model to utilize the activity groups for describing the behaviour of the apps at the activity level so that a high level of semantics can be compared to the API levels. Inactivity groups, a functioning cluster was adopted to characterize the behaviour for all the events. This is executed in such a way that the behaviour, concealed in every event, is still taken, and the EveDroid program is allowed to detect the novel malware at event levels. The authors developed a testing architecture named learning-based Android Malware Detection System

(TLAMD) to be applied in IoT gadgets [12]. The major concern lies in creating an appropriate fitness function that can generate efficient adversarial samples without impacting the feature of the applications.

Taheri et al. developed two defence methodologies against adverse attacks in malware detection process for mobile multi-media applications in an IoT environment [13]. In these methodologies, a powerful NN and a group of 1-nearest neighbour (C4N) and CNN were used. Then, the methods were trained using the dataset with adverse attacks. Consequently, the trained Machine Learning (ML) method achieved precise results. Further, when a malicious program arrives in the network through any other IoT device, the system triggers crucial warnings too. This study also described the attack model and algorithm for defending these attacks. Akbar et al. developed a Permission-based Malware Detection Method (PerDRaML) in which the App's malevolence was defined based on the application of the suspected permission [14]. Being a multi-level based model, this method identified and extracted a set of many characteristics, namely, permission rate, small size and permissions from the data gathered automatically-gathered from 10,000 applications. Furthermore, this study employed different ML models to classify the Apps as benign and malicious.

In the study conducted earlier [15], the authors proposed a malware detection technique to reduce the error rates and improve the accuracy by pre-processing and balancing the used datasets. To achieve this objective, static analysis was applied in this study to extract the features of the application. The feature ranking method was utilized in this study to pre-process the feature subset. Based on the ranks generated, the low-efficient features were eliminated. Also, the suggested method balanced the datasets using SMOTE, an under-sampling technique, along with a group of two other models which were not investigated earlier in the domain of detection techniques. Next, KNN, SVM, and Iterative Dichotomiser 3 classifiers were utilized to develop the recognition system. Niu et al. developed a novel method to categorize Android malware according to OpCode-level FCG and deep learning techniques [16]. The FCG was achieved with the help of static analysis of the Operation Code (OpCode), whereas this study used LSTM as its DL method.

3 The Proposed Model

This article has developed a new OGCNN-MDC model for malware detection and classification in IoT-cloud environment. The proposed OGCNN-MDC model aims to recognize and categorize the occurrences of malware in IoT-enabled cloud platforms. The presented OGCNN-MDC model has three stages: data pre-processing, malware detection and parameter tuning. Fig. 1 depicts the working processes of the OGCNN-MDC approach.

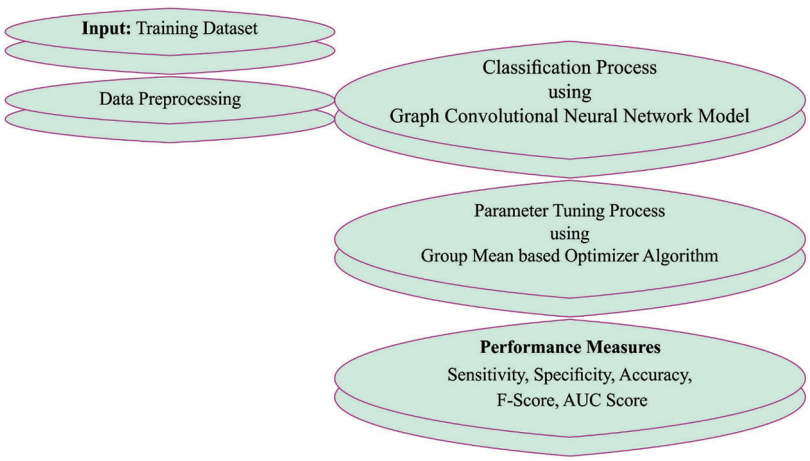


Figure 1: Working processes of the OGCNN-MDC approach

3.1 Malware Detection Using the GCNN Model

In order to detect and classify the malware, the GCNN model is exploited in this work. The GCNN approach implements the semi-supervised classifier [17]. The major concept is to upgrade the representation of the nodes by transmitting the data among the nodes.

Unlike the typical convolution that operates on a local Euclidean structure in the image, the GCNN approach aims to learn a function (.,.) on graph $\mathcal{G}$ that takes the feature description, $H^l \in \mathbb{R}^{n\times d}$ and the correlative matrix, $A \in \mathbb{R}^{n\times n}$ as input values (n represents the node count and d represents the dimensionality of the node feature) and upgrades the node feature as $H^{l+1} \in \mathbb{R}^{n\times d'}$.

$$H^{l+1} = f\left(H^l,\ A\right). \tag{1}$$

After applying the convolution function, $f(.,.)$ is denoted as follows.

$$H^{l+1} = h\left(\hat{A}H^l W^l\right), \tag{2}$$

In Eq. (2), $W^l \in \mathbb{R}^{d\times d'}$ indicates a conversion matrix that needs to be learnt, and $\hat{A} \in \mathbb{R}^{n\times n}$ shows the normalization form of the correlative matrix, A. Here, $h(.)$ signifies a nonlinear function performed by LeakyReLU in this study. Therefore, the complicated inter-relationships of the node can be modelled and learnt by stacking a different number of GCNN layers. The GCNN approach was developed for the classification of semi-supervised methods in which the node-level output is the predictive score of all the nodes. Unlike that, the authors developed a concluding output for all the GCNN nodes so that it acts as a class for all the respective labels in this process.

Furthermore, the graph model (viz., correlative matrixes) is generally pre-determined though it is not presented during the multi-label image detection process. Therefore, the correlative matrices should be developed from scratch. It encompasses two major components: GCNN-based classifier learning modules and image representation learning. Then, the CNN base model is applied to learn the features of the image. Therefore, an input image I with 448×448 resolution can develop as a $2048 \times 14 \times 14$ feature map from the conv5_x" layers. Next, the global max-pooling layer is employed to obtain the image-level feature, x.

$$x = f_{GMP}(f_{cnn}(I;\ \theta_{cnn})) \in \mathbb{R}^D, \tag{3}$$

In Eq. (3), θ_{cnn} denotes the model parameter and $D = 2048$.

$W = \{w_i\}_{i=1}^{C}$ is derived from label representation through a GCNN-based mapping function in which C signifies the category number. Then, the stacked GCNN is used in which l layer takes the node representation from the preceding layer $\left(H^l\right)$ as input, whereas the output is the new-fangled node representation, i.e., H^{l+1}. For the primary layer, the input is $Z \in \mathbb{R}^{C\times d}$ matrix. In this notation, d indicates the dimensionality of the label-level embedded word. For the previous layer, the output is $W \in \mathbb{R}^{C\times D}$, in which D denotes the dimensionality of the image demonstration. The prediction score can be determined from the learned classifier to image representation as

$$\hat{y} = Wx. \tag{4}$$

Consider that the ground truth label of the image is $y \in \mathbb{R}^C$ in which $y^i = \{0,\ 1\}$ denotes whether the label i appears in the image or not. Using conventional multi-label classification loss, the entire model is trained as given below.

$$\mathcal{L} = \sum\nolimits_{c=1}^{C} y^c \log\left(\sigma(\hat{y}^c)\right) + (1 - y^c) \log\left(1 - \sigma(\hat{y}^c)\right), \tag{5}$$

In Eq. (5), σ(.) represents the sigmoid function.

Typically, if a ‘surfboard’ occurs in an image, then a ‘person’ occurs with a high probability. But, due to the condition of the occurring ‘person’, the ‘surfboard’ not necessarily appears. Fig. 2 depicts the framework of the CNN method.

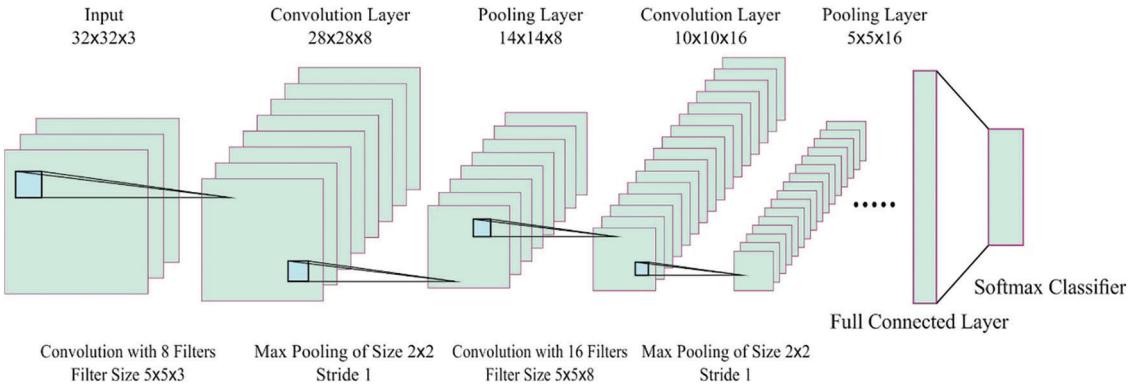


Figure 2: Structure of the CNN method

3.2 Hyperparameter Optimization Using GMBO Algorithm

In order to enhance the overall efficiency of the GCNN model, the GMBO algorithm is utilized to adjust the GCNN parameters appropriately. GMBO is a population-based optimization method proposed earlier based on efficiently using population member data when upgrading a model [18]. In all the iterations, two groups of members are chosen carefully, such as the bad group members and the good group members, with a specific number of members in each group. The foremost notion in the development of the presented method is to apply both integrated groups by averaging the number of two group members. The population member in the presented model is recognized using a matrix termed population matrix. The column count in the population matrix indicates the number of parameters in the problem whereas the row count in the population matrix designates the number of members in the algorithm. As a population member, every row of the population matrix is the solution proposed to resolve the optimization issue, which is determined as follows.

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix}_{N\times M} \begin{bmatrix} x_{1,1} & \cdots & x_{\mathrm{x}d} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ x_{i,1} & \cdots & x_{i,d} & \cdots & x_{i,m} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,d} & \cdots & x_{N,m} \end{bmatrix}_{N\times M} \tag{6}$$

Now, X denotes the population matrix, X_i shows the *i-th* member of the population, $x_{i,d}$ indicates the value of the *d-th* parameter that is generated using *i-th* member of the population, N shows the number of population members, and m indicates the number of parameters in the problem.

According to the variable value presented by every member of the population, the objective function is estimated. Therefore, the objective function value is defined by the vector as given below.

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix} \begin{bmatrix} F(X_1) \\ \vdots \\ F(X_i) \\ \vdots \\ F(X_N) \end{bmatrix}_{N\times 1}, \tag{7}$$

In Eq. (7), F denotes the vector of the main function, and F_i indicates the main function value, according to the *i-th* member of the population. Both good and bad group members are carefully chosen based on the objective function value. The good group has a specific number of population members with an optimal value of the main function. The bad group has a specific number of population members with the worst values of the objective function as shown below.

$$F^s = \begin{bmatrix} F_1^s \\ \vdots \\ F_i^s \\ \vdots \\ F_N^s \end{bmatrix} \begin{bmatrix} minimum\ (F) \\ \vdots \\ \vdots \\ \vdots \\ maximum\ (F) \end{bmatrix}_{N\times 1}, \tag{8}$$

$$X = \begin{bmatrix} X_1^s \\ \vdots \\ X_i^s \\ \vdots \\ X_N^s \end{bmatrix} \begin{bmatrix} x_{1,1}^s & \cdots & x_{1,d}^s & \cdots & x_{1,m}^s \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ x_{i,1}^s & \cdots & x_{i,d}^s & \cdots & x_{i,m}^s \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ x_{N,1}^s & \cdots & x_{N,d}^s & \cdots & x_{N,m}^s \end{bmatrix}_{N\times M}, \tag{9}$$

$$G_{N_G\times m} = X_i^s\ \&\ i = 1{:}N_{G'} \tag{10}$$

$$B_{N_B\times m} = X_i^s\ \&= N - N_B + 1{:}\ N, \tag{11}$$

In this expression, F^s denotes the sorted objective function vector, according to the objective function value arranged from the optimum member to the worst member, X^s indicates the sorted population matrixes, according to the value of the objective function, G denotes a certain good group, B denotes the bad group, N_G indicates the number of good groups and N_B denotes the number of bad groups. After defining both groups in this phase, two composite members are attained by averaging the values of the group as follows.

$$MG = mean(G_{N_G\times m}), \tag{12}$$

$$MB = mean(B_{N_B\times m}), \tag{13}$$

In this equation, MG and MB denote the composite members according to the mean values of the good and bad groups. In this work, the population matrix is upgraded through three phases, according to the best and the composite members as given below.

$$x_{i,d}^G = x_{i,d} + r \times \left(MG_{i,d} - x_{i,d}\right) \times sign(F_i - F_{MG}), \tag{14}$$

$$X_i = \begin{cases} X_i^B, & F_i^B < F_i \\ X_i, & else \end{cases}' \tag{15}$$

Now, $x_{i,d}^G$ denotes the novel value of the *d-th* parameter, r indicates an arbitrary value within [0–1], F_{MG} shows the objective function value of the composite members of the good groups, X_i^G represents the novel location of the *i-th* member of the population and F_i^G shows the value of objective function. Next, the population matrix is upgraded based on the composite members of the bad groups, as evaluated herewith.

$$x_{i,d}^B = x_{i,d} + r \times \left(MB_{i,d} - x_{i,d}\right) \times sign(F_i - F_{MB}), \tag{16}$$

$$X_i = \begin{cases} X_i^B, & F_i^B < F_i \\ X_i, & else \end{cases}' \tag{17}$$

Now, $x_{i,d}^B$ indicates the novel value of the *d-th* variable, r denotes an arbitrary value within [0—1], F_{MB} shows the objective function value of the composite members of the bad groups, X_i^B indicates the novel location of the *i-th* population member and F_i^B indicates the value of the objective function. Then, the population matrix is upgraded based on the optimum member of the population, as given below.

$$x'_{i,d} = x_{i,d} + r \times \left(x_{i,d}^{best} - x_{i,d}\right), \tag{18}$$

$$X_i = \begin{cases} X'_i, & F'_i < F_i \\ X_i, & else \end{cases}' \tag{19}$$

In this equation, $x'_{i,d}$ denotes the novel value of the *d-th* variable, r shows an arbitrary value within [0–1], X_i^i denotes the novel location of the *i-th* member of the population and F'_i represents the value of the objective function. The procedure of upgrading the population matrices is repeated until the process is satisfied. Then, during the final iteration, a quasi-optimal solution, i.e., output, is attained by the GMBO technique.

Algorithm 1: Pseudocode of the GMBO algorithm

Initialize GMBO.

1: Input problem data: constraints, variables, and objective function.

2: Determine the count of iterations (T) and population members (N).

3: Make a primary population matrix randomly.

4: Assess the objective function.

5: for $t = 1{:}\,T$

6: Sort population matrix related to Eqs. (8) and (9).

7: Upgrade good group related to Eq. (10).

8: Upgrade the bad group related to Eq. (11).

9: Upgrade composite members (MG and MB) related to Eqs. (12) and (13).

10: for $i = 1{:}\,N$

11: Upgrade date X_i depends on the primary phase utilizing Eqs. (14) and (15).

12: Upgrade X_i depends on the next phase utilizing Eqs. (16) and (17).

13: Upgrade X_i depends on the last stage utilizing Eqs. (18) and (19).

(Continued)

Algorithm 1 (continued)

14: end for

15: Save the optimum quasi-optimal solution attained with the GMBO so far.

16: end for

17: Resultant optimum quasi-optimal solution gained with the GMBO.

End GMBO.

4 Experimental Validation

This section discusses the malware classification results of the proposed OGCNN-MDC method in detail, while the experimental validation was conducted using a standard dataset. Table 1 gives the details of the dataset. The dataset holds a total of 33,269 samples under two classes.

Table 1: Dataset details

Class	No. of instances
Malware	5304
Benign	27965
Total no. of instances	33269

Fig. 3 illustrates the confusion matrices generated by the proposed OGCNN-MDC model during the classification process. With run-1, the proposed OGCNN-MDC model categorized 5,081 samples under the malware class and 27,764 samples under the benign class. Similarly, with run-2, the OGCNN-MDC approach categorized 5,064 samples as a malware class and 27,768 samples as a benign class. Also, with run-3, the proposed OGCNN-MDC technique classified 5,049 samples under the malware class and 27,778 samples under the benign class. Furthermore, with run-4, the proposed OGCNN-MDC technique placed 4,812 samples under the malware class and 27,843 samples under the benign class. Additionally, with run-5, the proposed OGCNN-MDC technique categorized 5,052 samples under the malware class and 27,774 samples under the benign class.

The overall cyberattack classification results of the proposed OGCNN-MDC model under distinct runs are given in Table 2.

Fig. 4 showcases the run-1 results of the proposed OGCNN-MDC model on both class labels. The figure denotes that the proposed OGCNN-MDC method proficiently identified both malware and benign classes. In malware class, the OGCNN-MDC model attained $accu_y$, $prec_n$, $sens_y$, $spec_y$, F_{score} and AUC_{score} values such as 98.73%, 96.19%, 95.80%, 99.28%, 95.99% and 97.54% respectively. Besides, in the benign class, the proposed OGCNN-MDC model reached $accu_y$, $prec_n$, $sens_y$, $spec_y$, F_{score} and AUC_{score} values such as 98.73%, 99.20%, 99.28%, 95.80%, 99.24% and 97.54% respectively. Also, the presented OGCNN-MDC model produced average $accu_y$, $prec_n$, $sens_y$, $spec_y$, F_{score} and AUC_{score} values such as 98.73%, 97.70%, 97.54%, 97.54%, 97.62% and 97.54% respectively.

Fig. 5 displays the run-2 results of the proposed OGCNN-MDC approach on both class labels. The figure is implicit that the OGCNN-MDC algorithm proficiently identified both malware and benign classes. In the malware class, the proposed OGCNN-MDC technique obtained $accu_y$, $prec_n$, $sens_y$, $spec_y$, F_{score} and AUC_{score} values such as 98.69%, 96.26%, 95.48%, 99.30%, 95.86% and 97.39%

correspondingly. In addition, in benign class, the presented OGCNN-MDC method reached $accu_y$, $prec_n$, $sens_y$, $spec_y$, F_{score} and AUC_{score} values such as 98.69%, 99.14%, 99.30%, 95.48%, 99.22% and 97.39% correspondingly. Likewise, the proposed OGCNN-MDC approach produced average $accu_y$, $prec_n$, $sens_y$, $spec_y$, F_{score} and AUC_{score} values such as 98.69%, 97.70%, 97.39%, 97.39%, 97.54% and 97.39% correspondingly.

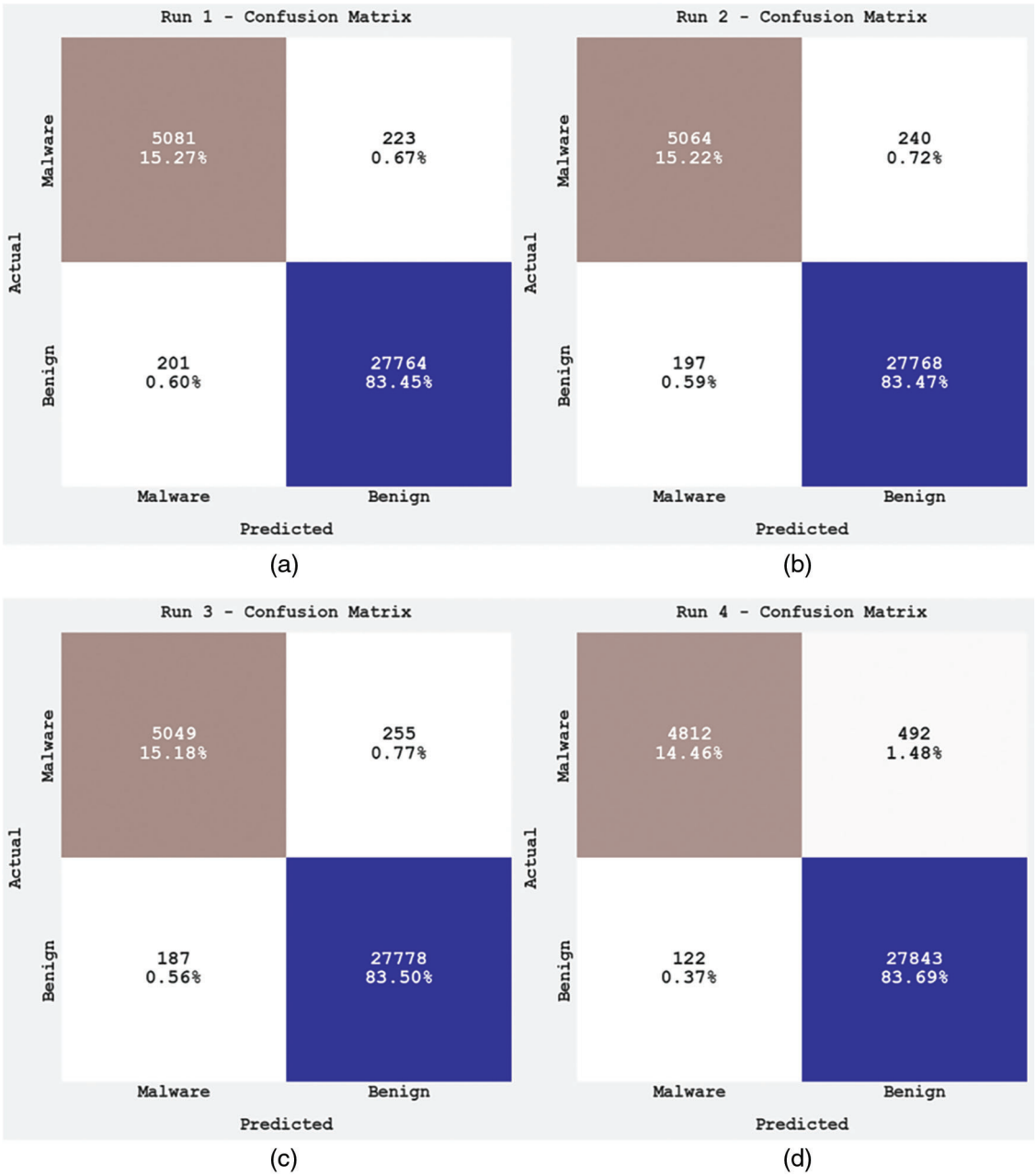
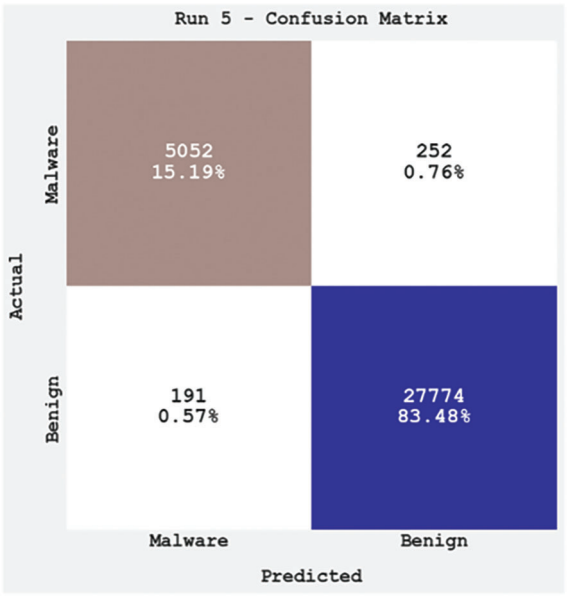


Figure 3: (Continued)



(e)

Figure 3: Confusion matrices of OGCNN-MDC approach (a) Run1, (b) Run2, (c) Run3, (d) Run4, and (e) Run5

Table 2: Analytical results of the OGCNN-MDC approach under different measures and runs

Labels	Accuracy	Precision	Sensitivity	Specificity	F-Score	AUC Score
Run-1						
Malware	98.73	96.19	95.80	99.28	95.99	97.54
Benign	98.73	99.20	99.28	95.80	99.24	97.54
Average	**98.73**	**97.70**	**97.54**	**97.54**	**97.62**	**97.54**
Run-2						
Malware	98.69	96.26	95.48	99.30	95.86	97.39
Benign	98.69	99.14	99.30	95.48	99.22	97.39
Average	**98.69**	**97.70**	**97.39**	**97.39**	**97.54**	**97.39**
Run-3						
Malware	98.67	96.43	95.19	99.33	95.81	97.26
Benign	98.67	99.09	99.33	95.19	99.21	97.26
Average	**98.67**	**97.76**	**97.26**	**97.26**	**97.51**	**97.26**
Run-4						
Malware	98.15	97.53	90.72	99.56	94.00	95.14
Benign	98.15	98.26	99.56	90.72	98.91	95.14
Average	**98.15**	**97.90**	**95.14**	**95.14**	**96.46**	**95.14**

(Continued)

Table 2 (continued)

Labels	Accuracy	Precision	Sensitivity	Specificity	F-Score	AUC Score
Run-5						
Malware	98.67	96.36	95.25	99.32	95.80	97.28
Benign	98.67	99.10	99.32	95.25	99.21	97.28
Average	**98.67**	**97.73**	**97.28**	**97.28**	**97.50**	**97.28**

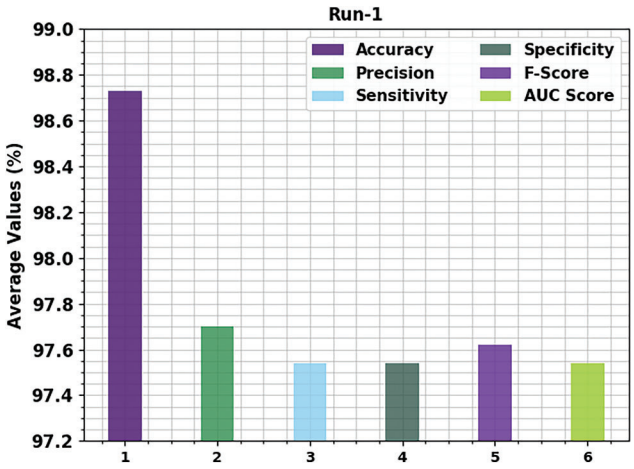


Figure 4: Average analysis results of the OGCNN-MDC approach under Run-1

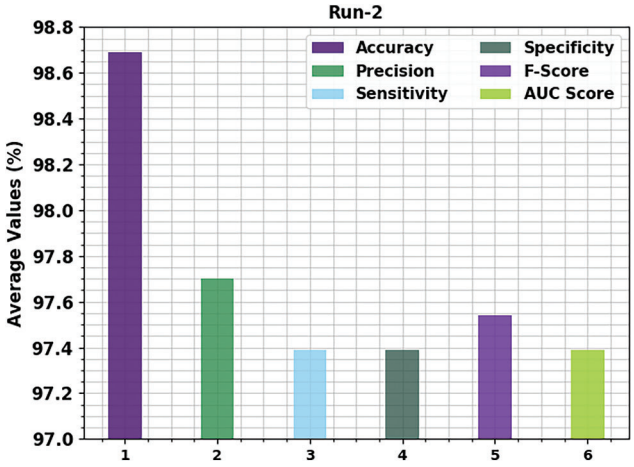


Figure 5: Average analysis results of the OGCNN-MDC approach under Run-2

Fig. 6 exemplifies the run-3 results of the proposed OGCNN-MDC approach on both class labels. The figure implies that the proposed OGCNN-MDC method identified both malware and benign classes excellently. In malware class, the proposed OGCNN-MDC technique gained $accu_y$, $prec_n$, $sens_y$, $spec_y$,

F_{score} and AUC_{score} values such as 98.67%, 96.43%, 95.19%, 99.33%, 95.81% and 97.26% correspondingly. In benign class, the proposed OGCNN-MDC approach achieved $accu_y$, $prec_n$, $sens_y$, $spec_y$, F_{score} and AUC_{score} values such as 98.67%, 99.09%, 99.33%, 95.19%, 99.21% and 97.26% correspondingly. Likewise, the proposed OGCNN-MDC approach produced average $accu_y$, $prec_n$, $sens_y$, $spec_y$, F_{score} and AUC_{score} values such as 98.67%, 97.76%, 97.26%, 97.26%, 97.51% and 97.26% correspondingly.

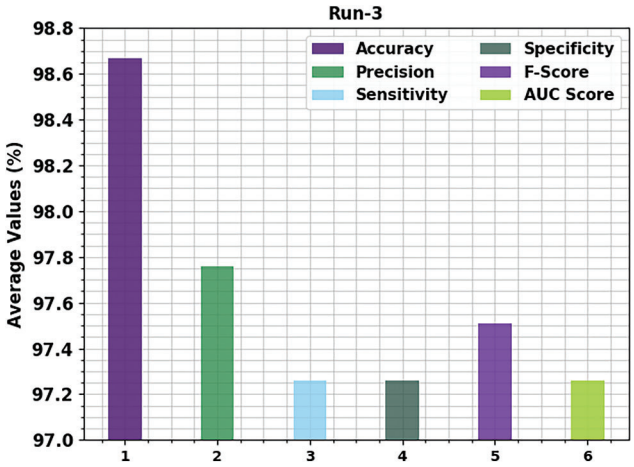


Figure 6: Average analysis results of the OGCNN-MDC approach under Run-3

Fig. 7 displays the run-4 results of the proposed OGCNN-MDC approach on both class labels. The figure infers that the proposed OGCNN-MDC method proficiently identified both malware and benign classes. In the malware class, the proposed OGCNN-MDC algorithm gained $accu_y$, $prec_n$, $sens_y$, $spec_y$, F_{score} and AUC_{score} values such as 98.15%, 97.53%, 90.72%, 99.56%, 94% and 95.14% correspondingly. In addition, in benign class, the presented OGCNN-MDC approach reached $accu_y$, $prec_n$, $sens_y$, $spec_y$, F_{score} and AUC_{score} values such as 98.15%, 98.26%, 99.56%, 90.72%, 98.91% and 95.14% correspondingly. Similarly, the proposed OGCNN-MDC method produced average $accu_y$, $prec_n$, $sens_y$, $spec_y$, F_{score} and AUC_{score} values such as 98.15%, 97.90%, 95.14%, 95.14%, 96.46% and 95.14% correspondingly.

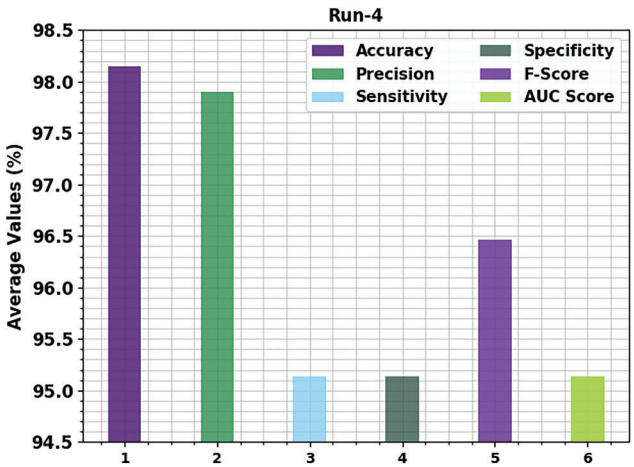


Figure 7: Average analysis results of the OGCNN-MDC approach under Run-4

Fig. 8 portrays the run-5 results of the OGCNN-MDC method on both class labels. The figure denotes that the proposed OGCNN-MDC approach proficiently identified both malware and benign classes. In malware class, the proposed OGCNN-MDC methodology gained $accu_y$, $prec_n$, $sens_y$, $spec_y$, F_{score} and AUC_{score} values such as 98.67%, 96.36%, 95.25%, 99.32%, 95.80% and 97.28% correspondingly. Also, in benign class, the proposed OGCNN-MDC methodology achieved $accu_y$, $prec_n$, $sens_y$, $spec_y$, F_{score} and AUC_{score} values such as 98.67%, 99.10%, 99.32%, 95.25%, 99.21% and 97.28% respectively. In addition, the proposed OGCNN-MDC methodology produced average $accu_y$, $prec_n$, $sens_y$, $spec_y$, F_{score} and AUC_{score} values such as 98.67%, 97.73%, 97.28%, 97.28%, 97.50% and 97.28% correspondingly.

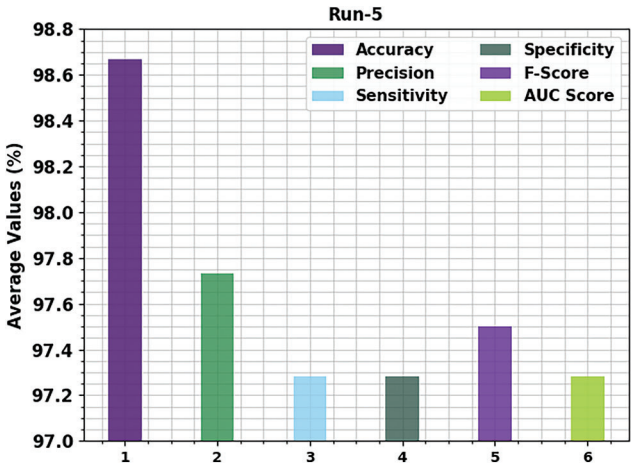


Figure 8: Average analysis results of the OGCNN-MDC approach under Run-5

Both Training Accuracy (TRA) and Validation Accuracy (VLA) values, acquired by the proposed OGCNN-MDC approach on the test dataset, are displayed in Fig. 9. The experimental results infer that the OGCNN-MDC method attained the maximal TRA and VLA values. In contrast, VLA values were higher than the TRA values.

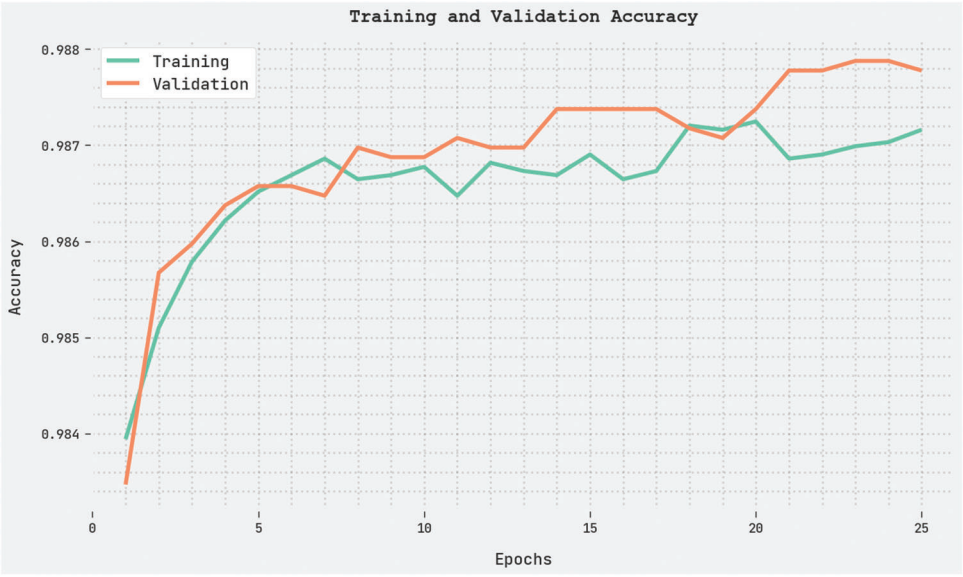


Figure 9: TRA and VLA analyses results of the OGCNN-MDC approach

Both Training Loss (TRL) and Validation Loss (VLL) values attained by the proposed OGCNN-MDC approach on the test dataset are exhibited in Fig. 10. The experimental results denote that the proposed OGCNN-MDC method exhibited the least TRL and VLL values. In contrast, the VLL values were lesser than the TRL values.

Figure 10: TRL and VLL analyses of the OGCNN-MDC approach

A clear precision-recall analysis was conducted upon the OGCNN-MDC method using the test dataset. The results are portrayed in Fig. 11. The figure shows that the proposed OGCNN-MDC technique produced enhanced precision-recall values under all the classes. A brief ROC analysis was conducted on the proposed OGCNN-MDC technique using the test dataset. The results are shown in Fig. 12. The results represent that the proposed OGCNN-MDC method established its ability to categorise the test dataset under distinct classes.

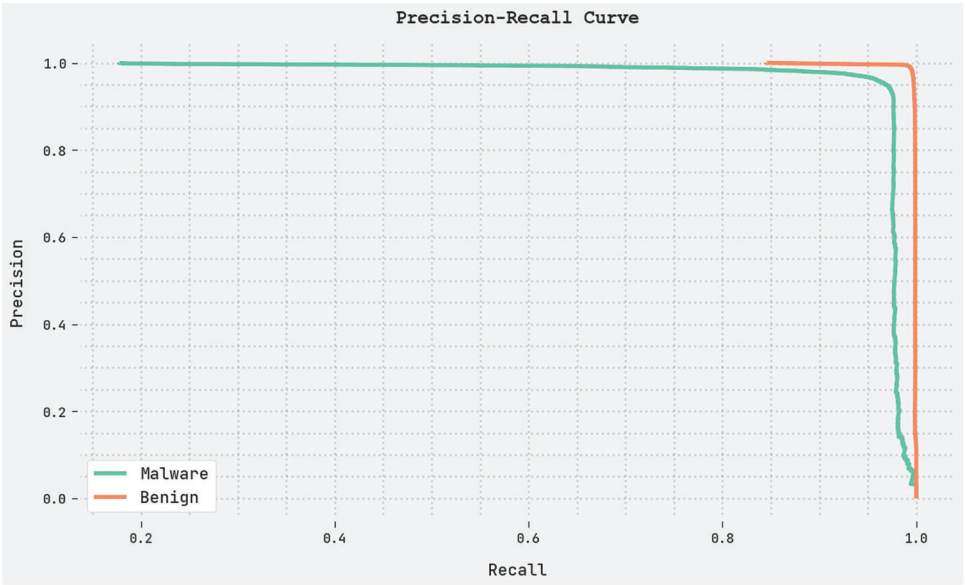


Figure 11: Precision-recall analysis results of the OGCNN-MDC approach

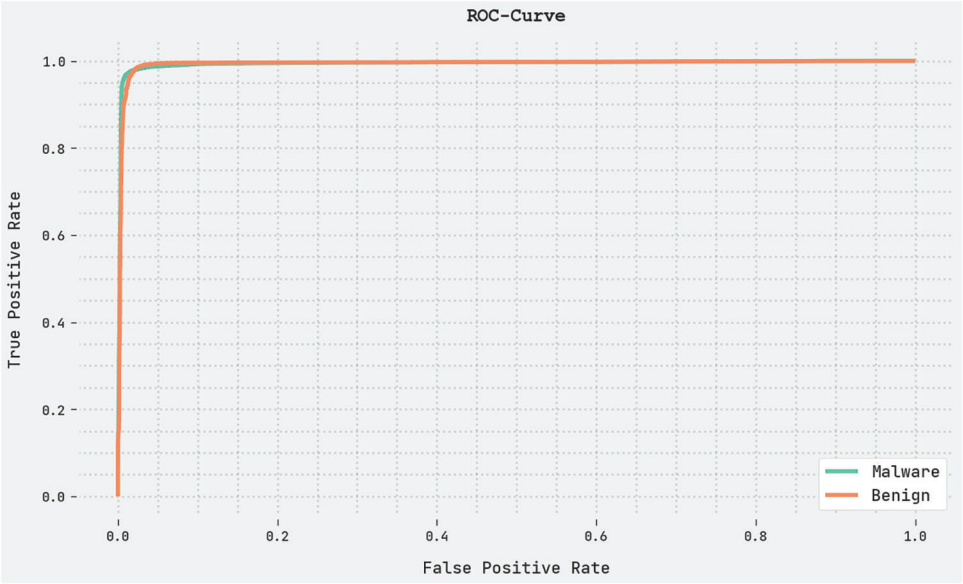


Figure 12: ROC analysis results of the OGCNN-MDC approach

At last, a brief comparative examination was conducted between the OGCNN-MDC method and other ML approaches and the results are provided in Table 3 [19]. Fig. 13 provides the comparative $accu_y$ investigation results achieved by the proposed OGCNN-MDC model and other ML methods. The figure indicates that the SGD and MLP models achieved ineffectual outcomes with minimal $accu_y$ values, such as 93.71% and 93.81%, respectively. Meanwhile, the NB and LR models revealed slightly enhanced outcomes with $accu_y$ values such as 94.85% and 94.76%, correspondingly. Furthermore, the RF and RT models managed to produce reasonable $accu_y$ values, such as 96.44% and 95.19%, respectively. Finally, the proposed OGCNN-MDC model achieved an effectual performance with a maximum $accu_y$ of 98.73%.

Table 3: Comparative analysis results of the OGCNN-MDC approach and other existing algorithms

Methods	Accuracy	Precision	Sensitivity	Specificity
OGCNN-MDC	98.73	97.70	97.54	97.54
Naïve bayes	94.85	93.98	96.05	93.27
Logistic regression	94.76	93.65	95.76	94.55
Stochastic gradient descent	93.71	93.52	93.88	94.69
Multilayer perceptron	93.81	93.58	93.81	95.62
Random forest	96.44	96.05	96.76	95.80
Random tree	95.19	96.66	93.38	93.60

Fig. 14 presents the detailed comparative $prec_n$ examination outcomes of the OGCNN-MDC approach and other ML models. The figure denotes that the SGD and MLP techniques achieved ineffectual outcomes with minimal $prec_n$ values such as 93.52% and 93.58%, correspondingly. Meanwhile, the NB and LR methods exposed slightly enhanced outcomes with $prec_n$ values such as 93.98% and 93.65%, correspondingly. Also, the RF and RT techniques yielded reasonable $prec_n$ values, such as 96.05% and

96.66%, correspondingly. At last, the proposed OGCNN-MDC approach accomplished an effectual performance with a maximum $prec_n$ of 97.70%.

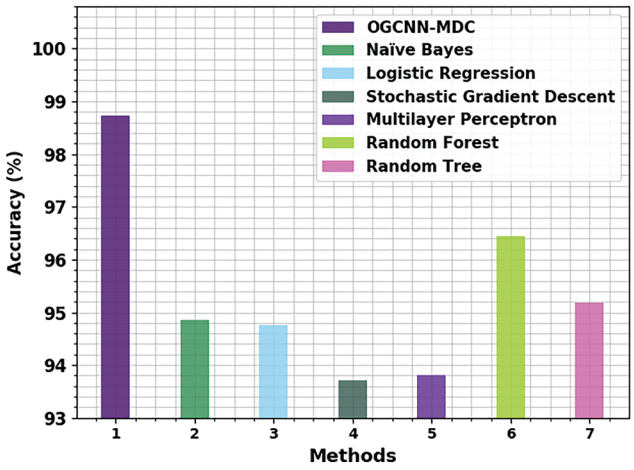


Figure 13: $Accu_y$ analysis results of the OGCNN-MDC approach and other existing algorithms

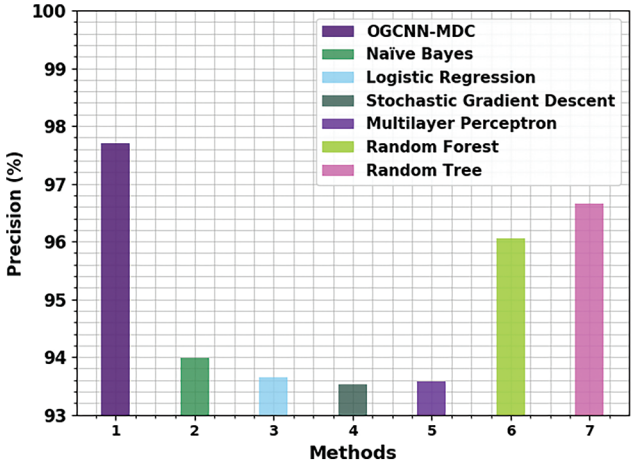


Figure 14: $Prec_n$ analysis results of the OGCNN-MDC approach and other existing algorithms

Fig. 15 illustrates the comprehensive $senc_y$ study outcomes achieved by the proposed OGCNN-MDC approach and other ML algorithms. The figure denotes that the SGD and MLP algorithms accomplished ineffectual outcomes with minimal $senc_y$ values, such as 93.88% and 93.81%, correspondingly. In the meantime, the NB and LR methods yielded slightly enhanced outcomes with $senc_y$ values such as 96.05% and 95.76%, correspondingly. Also, the RF and RT techniques produced reasonable $senc_y$ values, such as 96.76% and 93.38%, correspondingly. Finally, the proposed OGCNN-MDC approach accomplished an effectual performance with a maximum $senc_y$ of 97.54%.

Fig. 16 demonstrates the comparative $spec_y$ investigation outcomes of the proposed OGCNN-MDC model and other ML techniques. The figure indicates that the SGD and MLP methods achieved ineffectual outcomes with minimal $spec_y$ values such as 94.69% and 95.62%, correspondingly. In the meantime, the NB and LR methods produced slightly enhanced outcomes, with $spec_y$ values being 93.27% and 94.55% correspondingly. Besides, the RF and RT approaches yielded reasonable $spec_y$ values

such as 95.80% and 93.60%, respectively. At last, the proposed OGCNN-MDC technique displayed an effectual performance with a maximum $spec_y$ of 97.54%. Therefore, the proposed OGCNN-MDC model can be utilized to ensure cybersecurity in an IoT environment.

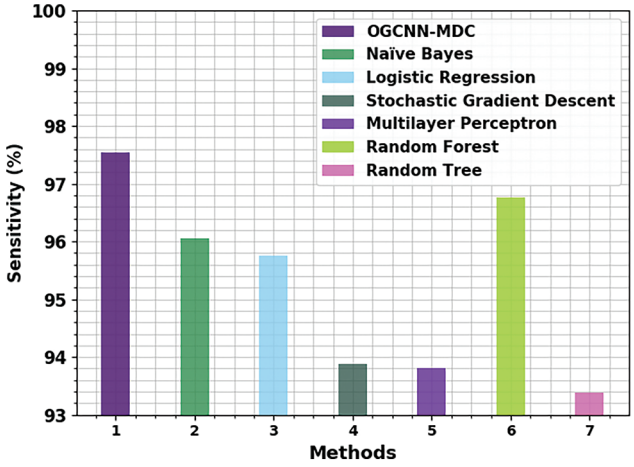


Figure 15: $Sens_y$ analysis results of the OGCNN-MDC approach and other existing algorithms

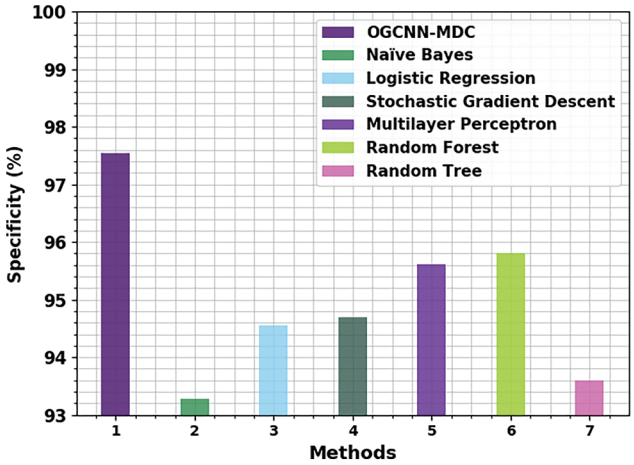


Figure 16: $Spec_y$ analysis results of the OGCNN-MDC approach and other existing algorithms

5 Conclusion

This article has developed a new OGCNN-MDC model for malware detection and classification in an IoT-enabled cloud environment. The proposed OGCNN-MDC model aims to recognize and categorize the occurrences of malware in the IoT-enabled cloud platform. The presented OGCNN-MDC model has three stages, namely, data pre-processing, malware detection and parameter tuning. To detect and classify the malware, the GCNN model is exploited in this work. To enhance the overall efficiency of the GCNN model, the GMBO algorithm is utilized to adjust the GCNN parameters appropriately. A widespread experimental analysis was conducted to highlight the improvements of the proposed OGCNN-MDC method. A comprehensive comparison study was conducted, and the outcomes confirmed the superiority of the proposed OGCNN-MDC method over other recent approaches with an accuracy of 98.73%.

Funding Statement: Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2022R237), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. The authors would like to thank the Deanship of Scientific Research at Umm Al-Qura University for supporting this work by Grant Code: (22UQU4331004DSR13).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Abawajy, A. Darem and A. A. Alhashmi, "Feature subset selection for malware detection in smart IoT platforms," *Sensors*, vol. 21, no. 4, pp. 1374, 2021.

[2] M. A. Omer, S. R. Zeebaree, M. A. Sadeeq, B. W. Salim, S. X. Mohsin *et al.,* "Efficiency of malware detection in android system: A survey," *Asian Journal of Research in Computer Science*, vol. 7, no. 4, pp. 59–69, 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 *et al.,* "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," *Journal of Internet Services and Information Security*, vol. 11, no. 2, pp. 34–45, 2021.

[7] H. Zhu, Y. Li, R. Li, J. Li, Z. You *et al.,* "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, pp. 10–22967, 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] T. Lei, Z. Qin, Z. Wang, Q. Li and D. Ye, "EveDroid: Event-aware android malware detection against model degrading for IoT devices," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6668–6680, 2019.

[12] X. Liu, X. Du, X. Zhang, Q. Zhu, H. Wang *et al.,* "Adversarial samples on android malware detection systems for IoT systems," *Sensors*, vol. 19, no. 4, pp. 974, 2019.

[13] R. Taheri, R. Javidan and Z. Pooranian, "Adversarial android malware detection for mobile multimedia applications in IoT environments," *Multimedia Tools and Applications*, vol. 80, no. 11, pp. 16713–16729, 2021.

[14] F. Akbar, M. Hussain, R. Mumtaz, Q. Riaz, A. W. A. Wahab *et al.,* "Permissions-based detection of android malware using machine learning," *Symmetry*, vol. 14, no. 4, pp. 718, 2022.

[15] D. T. Dehkordy and A. Rasoolzadegan, "A new machine learning-based method for android malware detection on imbalanced dataset," *Multimedia Tools and Applications*, vol. 80, no. 16, pp. 24533–24554, 2021.

[16] W. Niu, R. Cao, X. Zhang, K. Ding, K. Zhang *et al.,* "Opcode-level function call graph based android malware classification using deep learning," *Sensors*, vol. 20, no. 13, pp. 3645, 2020.

[17] Y. D. Zhang, S. C. Satapathy, D. S. Guttery, J. M. Górriz and S. H. Wang, "Improved breast cancer classification through combining graph convolutional network and convolutional neural network," *Information Processing & Management*, vol. 58, no. 2, pp. 102439, 2021.

[18] M. Dehghani, Z. Montazeri and Š Hubálovský, "GMBO: Group mean-based optimizer for solving various optimization problems," *Mathematics*, vol. 9, no. 11, pp. 1190, 2021.

[19] A. Fournier, F. El Khoury and S. Pierre, "A client/server malware detection model based on machine learning for android devices," *IoT*, vol. 2, no. 3, pp. 355–374, 2021.