IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 15, No. 1, February 2026, pp. 140~153
ISSN: 2252-8938, DOI: 10.11591/ijai.v15.i1.pp140-153

Malware detection using convolutional neural network-di strategy polar fox optimization algorithm

Parvathi Sathenahalli Jayaprakash[1,2], Yogeesh Ambalagere Chandrashekaraiah[1]
[1]Department of Computer Science and Engineering, Government Engineering College Chamarajanagar, Affiliated to Visvesvaraya Technological University, Belagavi, India
[2]Department of Information Science and Engineering, JSS Science and Technology University, Mysuru, India

Article Info

Article history:

Received Mar 20, 2025
Revised Nov 13, 2025
Accepted Jan 10, 2026

Keywords:

Cauchy operator mutation
Convolutional neural network
Di strategy polar fox optimization algorithm
Malware detection
Sine chaotic mapping

ABSTRACT

Malware attacks have escalated significantly with an increase of internet users and connected devices. With the rise of various types of malwares released by the hackers, constructing new competitive methods are necessary to identify the advanced malware. However, conventional malware detection struggles to identify new and evolving malware variants accurately because of its dependence on handcrafted features and static-signature based methods. To address this problem, this research proposes convolutional neural network (CNN) based di strategy polar fox optimization algorithm (DSPFOA) for malware detection to fine-tune the CNN parameters effectively which later assists to overcome the limitations of CNN. The model integrates the sine chaotic mapping and Cauchy operator mutation as DSPFOA prevents the model from local optima issue, and extends search space solution, also enhance convergence. This ensures that the CNN learns highly discriminative features which makes the system more accurate and robust in detecting both known and evolving malware variants. The CNN-DSPFOA achieves a high accuracy of 99.65 and 99.76% by utilizing BIG2015 and Malimg dataset respectively compared to existing methods like masked self-supervised model with swin transformer (MalSort).

Corresponding Author:

Parvathi Sathenahalli Jayaprakash
Department of Computer Science and Engineering, Government Engineering College Chamarajanagar
Affiliated to Visvesvaraya Technological University
Belagavi, India
Email: sjparvathi25@gmail.com

1. INTRODUCTION

Malware is a malicious program that harms and invades the computing devices typically by compromising the protocols which include threats such as Trojan horses, viruses, adware, worms, ransomware, rootkits, botnets, and spyware [1]. Malware is often used to steal information, utilize hardware to disrupt for reputational or financial gain, or other unauthorized activity. Malware defense involves numerous layers, including altering users when a system is compromised, preventing malware infiltration, and removing malware from infected systems [2]. In recent years, the number of cybercrimes and breaches have significantly increased, especially in Windows and internet of things (IoT) environments [3].

LokiBot which is a malicious software is designed to steal cryptocurrency wallets, passwords, and other sensitive information by placing malicious code into files that are cloud and hosted-based services [4]. Malware detection approaches are typically based on dynamic analysis or static analysis. The static do not contain the actual execution of a program, which has control flow graphs and opcode sequences [5]. But the dynamic analysis executes the program based on virtual environments. It assists in managing whether the

application programming interface (API) is being called, system calls are executed, instructions are traced, registry changes occur, or memory is modified [6]. However, both dynamic and static analyses failed due to attackers having determined various methods for evading detection. For instance, a static malware analysis evaluates the executable file that is bypassed by a well-disguised malware. In addition, malware can bypass the detection in dynamic analysis by simply altering its behavior. All these circumstances led to the determination of new effective methods for malware detection [7].

To solve malware attacks, a signature-based malware detection system application, like an anti-virus program, is established based on the signature database which is extracted from early determined malware samples that become popular. It compares file characteristics to known signatures and evaluates only if a match is determined to identify and prevent malware threats [8], [9]. Moreover, signature-based malware also has a limitation, where it does not modify or identify new threats and is computationally intensive because of processing a huge amount of data. A malware family is a collection of malware samples with the base of a same code [10], [11]. Accordingly, each malware family contains its visual similarities and properties, which are different from other malware families. A malware writer tries to contaminate the training data; hence, the resulting approach is less efficient. A possible method for such an adversarial attack on an image-based malware detection is for generating "deep fake" images to pollute the training data. Therefore, deep learning (DL) provides numerous benefits over conventional machine learning (ML) [12], [13], such as automatic generation of high-quality features and the ability to manage large data effectively [14].

Wang *et al*. [15] presented a masked self-supervised model with swin transformer (MalSort) to classify malware effectively. Initially, each malware instance was converted into color image, and then swin transformer was applied to extract the multi-scale key feature vectors. At last, an encoder was fine-tuned to perform malware classification efficiently using MalSort. However, the MalSort struggled with fine-grained malware classification as it relied on the self-supervised learning that overlooked subtle variations in malware patterns. Liu *et al*. [16] established a convolutional neural network (CNN) for multi-class malware detection. The balanced sampling and augmentation methods were employed to solve the imbalance issue, which ensured generalization ability. Then, the gray level co-occurrence matrix (GLCM) was used to extract the features from malicious code feature maps. The outcomes focused on the model accuracy for each data recognition and also considered the recognition effect of each malware classes. Nevertheless, CNN struggled to detect malware effectively because of its limited ability to capture fine-grained differences among subtle variants of malware.

Mosleh and Sharifian [17] suggested a hierarchical cloud deep neural network (DNN) for malware classification in IoT. The suggested model was effectively scaled from IoT devices to cloud and edge for enhancing the malware detection and simultaneously managing the accuracy level and minimizing resource utilization. The hierarchical cloud DNN effectively reduced both resource consumption and run-time also managed the performance characteristics. However, hierarchical cloud DNN faced challenges in scalability issues because of multi-layered processing across cloud nodes that led to delayed threat detection. Shaukat *et al*. [18] developed a hybrid model by integrating ML and deep transfer learning for malware detection. Initially, the deep transfer learning method was employed for extracting each deep feature from the fully connected layer of the DL model. Then, the ML was utilized as a final detector, which fully employed the inherent associations among input and output. The developed approach eliminated the necessity for knowledge from domain experts for inverse engineering performance. The developed hybrid model was cost-effective, scalable, and efficient in malware detection. Nevertheless, the developed approach suffered from feature redundancy because of overlapping extracted features, which resulted in minimized effectiveness of the model.

Singh *et al*. [19] introduced a multi-level feature extraction to categorize malware families. Initially, significant features from malware images were extracted by using gated recurrent unit (GRU), which were passed through CNN for the final feature vector extraction. At last, numerous malware families were classified by employing cost-sensitive boot strapped weighted random forest (CSBW-RF), which enhanced the model performance. However, the multi-level feature extraction struggled in capturing subtle and evolving features across new malware variants that resulted in overfitting or poor generalization. Kim *et al*. [20] established a cross-modal attention mechanism with CNN to classify the malware effectively. The malware images and structural entropies were the two modalities which were transformed and extracted from binary files. Both modalities contained diverse granularities of chunks and bytes which complemented each other. To integrate two modalities' features, cross-modal attention model was used which enhanced the model performance. Table 1 shows the description of literature survey with strengths, limitations, and metrics.

However, conventional malware detection struggled in identifying new and evolving malware variants accurately due to its dependence on handcrafted features and static-signature based methods. Moreover, malware detection is an artificial intelligence (AI)-driven pattern recognition and generalization task where models are necessitated for learning discriminative features that distinguished malicious behavior from benign. To solve this limitation, CNN based di strategy polar fox optimization algorithm (DSPFOA) is proposed to detect the malware effectively by removing the dependence of handcrafted features and static

signatures. CNN employed adaptive intelligence mechanism for an automatic extraction of hierarchical features and then fine-tuning the parameters using DSPFOA which ensured better adaptability to new and evolving variants. The novelty of this research is CNN's feature learning with an adaptive ability of DSPFOA which enables accurate malware detection. Overall, by integrating DL with metaheuristic optimization, system generalizes across different malware families that solves the dynamic nature of modern cyber threats. The major contribution of this research is in traditional polar fox optimization algorithm (PFOA), the sine chaotic mapping and Cauchy behavior mutation were used as di-strategies for initial population and solved local optima to fine-tune the CNN parameter's values effectively. CNN's convolutional layer minimized the number of parameters, which resulted in rapid processing and memory usage that was appropriate for detecting malware systems. Min-max normalization was used to ensure that all features are scaled with a consistent range that improved model accuracy and effectiveness during training.

This research paper is organized as follows: section 2 shows a detailed explanation of the proposed method. Section 3 presents CNN-DSPFOA. Section 4 illustrates results and discussion. Finally, section 5 demonstrates the conclusion of the paper.

Table 1. Summary of existing malware detection methods with strength, limitations, and evaluation metrics

Author	Methods	Strength	Limitation	Metrics
Wang *et al.* [15]	MalSort	MalSort employs pre-trained encoder for fine-tuning which perform malware classification efficiently	MalSort struggled with fine-grained malware classification because it relied on self-supervised learning that overlooked subtle variations in malware patterns	Accuracy, recall, F1-score, precision
Liu *et al.* [16]	ConvNet	The outcomes not only focused on model accuracy for each data recognition but also considered the recognition effect of all malware classes.	CNN struggled with detecting malware effectively because of its limited ability to capture fine-grained differences among subtle variants of malware.	Accuracy
Mosleh and Sharifian [17]	Hierarchical CloudDNN	This method effectively reduces both resource consumption and run-time while managing performance characteristics	Hierarchical cloud DNN face challenges in scalability issues because of multi-layered processing across cloud nodes that leads to delayed threat detection	Accuracy, recall, F1-score, precision
Shaukat *et al.* [18]	Hybrid model	Hybrid model was cost-effective, scalable, and efficient in detecting the malware.	Hybrid model suffered from feature redundancy because of overlapping extracted features, which resulted in minimized model effectiveness.	Accuracy, recall, F1-score, precision
Singh *et al.* [19]	GRU+CNN+ RF	Numerous malware families were classified by employing CSBW-RF which enhances the model performance.	multi-level feature extraction has difficulty in capturing subtle and evolving features across new malware variants that result in overfitting or poor generalization	Accuracy, recall, F1-score, precision
Kim *et al.* [20]	Attention-based cross model CNN	To integrate two modalities' features, cross-modal attention model was used which enhance model performance	This method struggles with accurately identifying new and evolving malware variants	Accuracy, F1-score

2. METHOD

This research proposes CNN-DSPFOA for an effective malware detection. Initially, the BIG2015 and Malimg dataset are performed for evaluating the model performance. A color map module is applied to convert binary files into colorful images. Then, min-max normalization is used to pre-process the images and then CNN-DSFOA is performed for malware detection by fine-tuning the CNN parameter values. Figure 1 demonstrates the block diagram of the proposed CNN-DSPFOA.

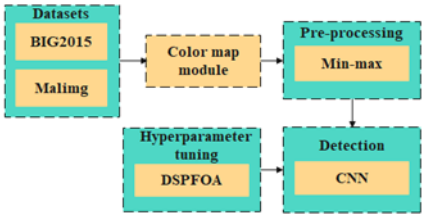


Figure 1. Block diagram illustrating the workflow of proposed CNN-DSPFOA method

2.1. Dataset

In this research, the BIG2015 [21] and Malimg [22] dataset are used to determine the performance of the model in malware. These datasets cover wide variety of malware families, which enhances the model's ability to generalize across different types. A detailed explanation of this dataset is described as follows:

i) BIG2015: it was published by Microsoft in 2015 with 9 malware families and 10,868 malware files. The files in this dataset are transformed into arrays and placed into memory for generating colored malware images. By converting malware data into structured format enables to determine malware patterns against threats.

ii) Malimg: it has various imbalanced Windows portable executable (PE) images which are adopted to classify malware family. It contains 9,339 malware samples that belong to 25 malware families. A pixel value of grayscale image is evaluated to all three RGB channels of a colorful image for generating malware colorful image. Then the dataset is split into 80% training and 20% testing, respectively and these obtained codes are fed into color-map module to generate colorful image.

2.2. Color-map module

After obtaining the code, the color-map module [15] is used to convert binary files containing malware into colorful images, which is essential due to its ability enhancement in DL to distinguish between patterns and subtle features. An image-based malware representation makes CNN to capture byte-level and structural patterns that are consistent in malware families and enhance the classification accuracy. This method employs visual similarities over malware variants which make effective feature extraction and classification. Color maps introduce dimension, which makes it easier to differentiate and highlight the malware families by varying intensities across RGB channels. The generation of colorful mapping is attained by using binary files or grayscale images that is related to malware. Initially, the malware binary files are converted into byte arrays to produce a colorful image and then stored in a memory buffer. This procedure is used for transforming BIG2015 dataset into colorful images whereas Malimg dataset has grayscale images resultant from binary files. A colorful image is established by assigning a grayscale image's pixel value to each channel of RGB. Figure 2 shows the process of colorful image and this representation assists in visual analysis and provides clearer insights into malware structure. Then, min-max normalization is utilized for normalizing image's pixel value to ensure consistency and improve the training process.

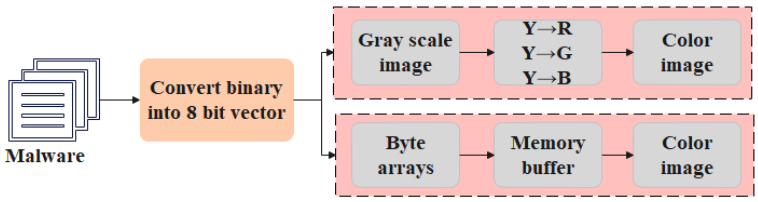


Figure 2. Visualization of color-map module utilized in malware image representation

2.3. Pre-processing

After the color code module, the min-max normalization scales the pixel values of the transformed malware images to a fixed range [0, 1]. This assists in solving the issues which are caused by a varying image intensity and ensures that the models treat every features equally. Min-max [23] provides relationships among original data values to manage the image's original shape which minimize the outlier impact. The mathematical formula for min-max normalization is expressed in (1) and (2).

$$X_{std} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (1)$$

$$X_{scaled} = X_{std} \times (max - min) + min \quad (2)$$

Where min represents the minimum feature range of an input data X and max denotes the maximum feature range respectively. Min-max enhance the training process and model's stability by minimizing the sensitivity to input variations. The pre-processed images are then passed through the CNN for malware detection.

3. CNN WITH DI-STRATEGY POLAR FOX OPTIMIZATION ALGORITHM

After pre-processing, CNN is applied for malware detection due to its ability to extract spatial and hierarchical patterns from malware binaries which are represented as images. CNN [24] automatically learn features without the requirement of manual feature engineering that enhances the detection accuracy. The convolutional layer of CNN effectively captures the structural similarities and anomalies within malware families, which makes robust classification. A detailed explanation of CNN is provided and its architecture for malware detection and classification process is shown in Figure 3.

i) Convolutional layer: convolution is generalized into numerous dimensions where the feature matrix of the filter g, and image f is defined in an integer set t with two input dimensions $f(c,d)$ c and d represents the width and height coordinates. The kernel size refers to the filter dimension which is employed in a convolution layer for extracting local features from pre-processed input. The mathematical formula for convolution output is expressed in (3) and (4). The zero paddings typically fit magnitude and hence, that spatial output dimension has a similar spatial input size.

$$h(c,d) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(p,q)g(c-p,d-q)dpdq \tag{3}$$

$$h(c,d) = \sum_p \sum_q f[p,q]g[c-p,d-q] \tag{4}$$

ii) Rectified linear unit (ReLU) activation: it assists in solving the vanishing gradient issue that allows deep networks to learn effectively which is formulated in (5). If the input is less than zero, then ReLU obtains an output value of zero; however, if other than zero, then output shows raw data. If the data is greater than zero, then the production is similar as input.

$$ReLU(h) = \begin{cases} 0 & if h < 0 \\ x & if\ h \geq 0 \end{cases} \tag{5}$$

iii) Pooling layer: it is used to ensure the output variance y with the integration of the convolutional layer. After the pooling layer, the network output is denoted using (6). The n_{out} represents the length or the height of an output and S indicates stride. The pooling process computes a statistical summary of the nearest data by utilizing an arithmetic function.

$$n_{out} = \frac{n_{in}+2P-F}{S} + 1 \tag{6}$$

iv) Fully connected: after convolutional and pooling layers, the fully connected layer is applied, which contains numerous neurons where each neuron is associated with all the neurons in adjacent layers. Those layers at the end of the network are employed to make detection and generate a final feature non-linear combination. The categorical cross entropy loss function is used in CNN method to minimize error. The techniques utilized to minimize the errors are collectively known as regularization, which is used as weight decay and dropout, respectively.

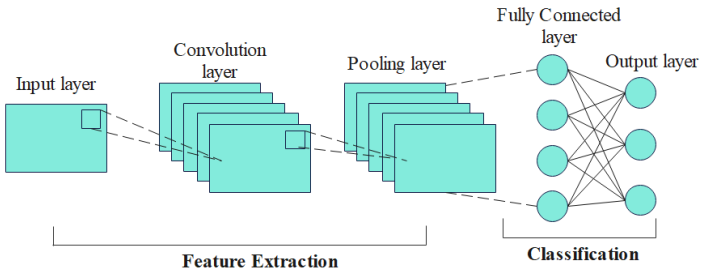


Figure 3. Architectural overview of designed CNN structure for malware detection and classification process

3.1. Hyperparameter tuning

In CNN, hyperparameter tuning is applied using DSPFOA, which is essential to optimize model performance that ensures high robustness and accuracy against the evolving threats. Effective tuning of the

parameters such as epochs, learning rate, optimizer, activation function, batch size, and dropout enhance the convergence speed. Without tuning, the model underperforms or overfits, which results in poor generalization on new malware samples. Optimized hyperparameters enhance the classification performance and minimize false positive (FP) and false negative (FP), respectively, in detecting the malware. Compared to existing methods like red fox optimization (RFO), fennec fox optimization (FFO), and artic fox optimization (AFO), the PFOA provides strong adaptability in harsh environments, which ensures an effective balance between exploration and exploitation. Its dynamic movement strategies make it escape local optima and enhance global search abilities. Its group coordination enhances decision-making in high dimensional and complex issues which makes PFOA a robust solution for a wide range of applications.

i) Initialization: PFOA is a nature-inspired metaheuristic that mimics the adaptive hunting and survival strategies of polar foxes to optimize intricate issues. The polar fox's initial population is generated randomly, which made the solution space using (7) and (8). The $LB = [lb_1 \quad lb_2 \quad \cdots \quad lb_d]$ and $UB = [ub_1 \quad ub_2 \quad \cdots \quad ub_d]$, X indicates polar fox's position matrix, x_j^i represents i^{th} polar fox with j^{th} value, k determines the number of polar foxes, and d illustrates dimension, $\overrightarrow{r_1}$ denotes a random vector in [0, 1] range, UB and LB demonstrates upper and lower bound respectively. The mean square error (MSE) is used as a fitness function in DSPFOA because it effectively measures the difference among predicted and actual outputs that guides the optimizer to reduce prediction errors.

$$X = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_d^1 \\ x_1^2 & x_2^1 & \cdots & x_d^2 \\ \vdots & \vdots & \cdots & \vdots \\ x_1^k & x_2^k & \cdots & x_d^k \end{bmatrix} \tag{7}$$

$$x^i = LB + \overrightarrow{r_1}.(UB - LB) \tag{8}$$

ii) Grouping polar fox: in each group, the number of members is similar; however, based on weight calculated for each group, the result is the failure or success of that group. Each group weight is updated using (9). The W_i depicts i^{th} group weight, NG_i the number of polar foxes in i^{th} group, and t indicates the present iteration. The group weight has an initial value that minimizes the rate of change.

$$W_i^{new} = W_i + \frac{t^2}{NG_i} \tag{9}$$

iii) Experience-based phase: the polar fox does not hibernate during winter; it exhibits an integration of communal and nomadic behavior for searching food. In (10) and (11) is applied for simulating the polar fox jumping behavior during hunting with the power of jump P and direction D by changing the position $x^i(t)$ to new one $x^i(t+1)$ using (10). This process is repeated till the subsequent condition is true: one is when the energy polar fox is less than set level, and another is when a better fitness is acquired, as shown in (11). The PF_i indicates i^{th} polar fox experimental power factor and f represents fitness.

$$x^i(t+1) = x^i(t) + P.D \tag{10}$$

$$\begin{cases} PF_i < m.PF \\ f(t) < f(t-1) \end{cases} \tag{11}$$

iv) Leader based phase: the polar fox each leash has a leader which leads the leash to obtain its goal. The position of leader L is based on optimal fittest in leash and the polar fox changes its position $x^i(t)$ to new position $x^i(t+1)$ using (12). The LF_i represents the leader power factor and $\overrightarrow{r_4}$ indicates random vector.

$$x^i(t+1) = x^i(t) + \overrightarrow{r_4}(x^i(t) - L).LF_i \tag{12}$$

v) Leader motivation phase, mutation and fatigue simulation: a leader motivates and members update their positions randomly while the polar foxes struggle to determine the prey consecutively by setting $G1_m$, $G2_m$, $G3_m$, and $G4_m$ as matrix behavior. This results in an increase in the process for a limited number of steps after the position is changed. The mathematical equation for the leader motivation phase is expressed in (13). Parents sometimes abandon litter of pups and dominant kits exhibit aggression toward their siblings. After leader motivation, the polar fox gets tired at each iteration and its efforts are minimized by $G1_r$, $G2_r$, $G3_r$, and $G4_r$ [25].

$$critical = (NLM > MLM) or (t > 0.8 \times NI) \tag{13}$$

3.1.1. Improved strategies

In traditional PFOA, the sin chaotic mapping is included in the population initialization stage for improving even distribution. This assists in enhancing the exploration abilities and avoiding premature convergence. Then, the Cauchy operation is incorporated to avoid local optima issues early on and increase the population search space. A detailed explanation for these improved strategies is explained as follows.

i) Sin chaotic mapping: it is used to initialize the population, which is higher level of chaotic behavior compared to logistic mapping. Incorporating a sine chaotic map during the PFOA initialization phase ensures more uniform distribution of population using (14). By using this strategy, the population is initialized, which results in more evenly distributed PFOA which enhances the model's performance and convergence speed.

$$\begin{cases} y_{n+1} = sin\frac{2}{y_n}, & n = 0,1,\dots,N \\ -1 < y_n < 1, & y_n \neq 0 \end{cases} \tag{14}$$

ii) Cauchy operator mutation: Cauchy distribution minimized slowly on both sides of the peak value, and polar fox minimizes the local optima issue after mutation. To increase the search process of PFOA, the Cauchy operator mutation is applied and the mathematical equation of 1 dimensional Cauchy function is expressed in (15). While $\delta = 1, \mu = 0$ then, the formula is represented in (16) and mathematical formula for conventional Cauchy distribution is indicated in (17).

$$f(y,\delta,\mu) = \frac{1}{\pi}\frac{\delta}{\delta^2+(y-\mu)^2}, -\infty < y < \infty \tag{15}$$

$$f(y,\delta,\mu) = \frac{1}{\pi}\frac{1}{y^2+1}, -\infty < y < \infty \tag{16}$$

$$Cauchy(0,1) = tan[(\xi - 0.5)\pi], \xi \epsilon U[0,1] \tag{17}$$

By integrating the position update of PFOA and Cauchy operation variation, the mutant individual is generated using (18). The updated individual is mutated by including a Cauchy operator randomly in each dimension to solve the local optima issue. Where β represents the disturbance factor. By incorporating these two strategies, the population distribution is presented as more uniform in the initialization stage, which increases convergence speed. The Cauchy operator solves the local optima issue and extends the search space of the population effectively in PFOA.

$$Y_{new(y)} = Y_i + \beta . Cauchy(0,1) \tag{18}$$

Table 2 shows the hyperparameter values of the proposed method to ensure reproducibility. The values are chosen based on grid search with 50 epoch for convergence, a batch size of 256 and 50 epochs provide stable gradient updates and sufficient learning iterations of 100. A learning rate of 0.001 with Adam optimizer ensures steady and adaptive convergence, ReLU activation function enhance learning with solving vanishing gradients, and dropout rate of 0.5 avoids overfitting by deactivating half of the neurons during training which prevents network from relying too heavily on specific features. Algorithm 1 shows a pseudocode for proposed method to ensure reproducibility.

Algorithm 1. Pseudocode of proposed CNN-DSPFOA for malware detection
Input: Dataset {BIG2015, Malimg}
Output: Predicted labels
Step 1: Data pre-processing
  i) For each sample x in X:
    – Convert malware binary into image
    – Apply min-max normalization
  ii) Split dataset into training, testing, and validation
Step 2: Initialize CNN method:
  i) Define convolutional layers with strides, kernel sizes, and activation function
  ii) Define pooling and FC layers
Step 3: Initialize DSPFOA
  i) Initialize population of candidate CNN hyperparameters
  ii) Apply max iterations

Step 4: Optimize CNN hyperparameters by utilizing DSPFOA
For iteration=1 to max iterations:
For each candidate solution in population:
i) Train CNN on training data by utilizing candidate hyperparameters
ii) Calculate fitness=MSE
iii) Update candidate solution by employing sine chaotic map for initialization and Cauchy mutation to solve local optima
iv) Choose the optimal candidate solution depending on lowest MSE
Step 5: Train final CNN for classification:
i) Apply CNN hyperparameters to be optimal solution determined by DSPFOA
ii) Train CNN on full training set
iii) Predict labels
End

Table 2. Summary of optimized hyperparameter values employed in proposed CNN-DSPFOA

Parameters	Values
Epochs	50
Batch size	256
Learning rate	0.001
Optimizer	Adam
Activation function	ReLU
Dropout	0.5

4. RESULTS AND DISCUSSION

The CNN-DSPFOA is simulated using a Python 3.4 environment with an intel i7 processor, Windows 10 operating system, and 64 GB RAM. F1-score, recall, accuracy, and precision are employed to determine the model performance using (19) to (22). The TP demonstrates true positive and TN indicates true negative.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{19}$$

$$Recall = \frac{TP}{TP+FN} \tag{20}$$

$$Precision = \frac{TP}{TP+FP} \tag{21}$$

$$F1-score = \frac{2TP}{2TP+FP+FN} \tag{22}$$

4.1. Performance analysis

Table 3 demonstrates the evaluation of different detection and classification techniques on BIG2015 and Malimg datasets. The existing techniques such as ResNet, vision transformer (ViT), swin transformer, and graph transformer are compared with CNN. When compared to these methods, the CNN obtains a high accuracy of 99.65±0.096 and 99.76±0.056 using BIG2015 and Malimg dataset due to its ability to extract hierarchical spatial features from malware images.

Table 3. Performance evaluation results of different malware detection and classification techniques on BIG2015 and Malimg datasets

Methods	Dataset	Accuracy (%)	Recall (%)	Precision (%)	F1-score (%)	t-test from p-value	Confidence interval (%)
ResNet	BIG2015	85.69±0.156	81.29±2.365	85.29±2.015	83.24±1.857	0.036	86.12
ViT		89.32±1.258	79.43±1.287	88.11±3.158	83.54±2.390	0.032	88.06
Swin transformer		92.91±1.356	76.31±0.236	87.43±2.784	81.49±1.258	0.029	90.78
Graph transformer		95.36±2.875	94.08±0.458	93.78±1.236	93.92±2.364	0.027	91.50
CNN		99.65±0.096	98.09±0.148	98.97±0.058	98.52±0.1023	0.025	93.26
ResNet	Malimg	89.43±0.365	92.38±2.354	92.19±1.487	92.28±2.158	0.038	87.63
ViT		93.29±0.254	94.50±1.087	91.28±1.458	92.86±3.784	0.034	89.48
Swin transformer		95.39±0.357	95.38±0.235	89.3l±0.148	90.43±2.791	0.031	91.38
Graph transformer		96.28±1.369	95.89±0.185	96.02±0.358	95.95±3.496	0.030	91.35
CNN		99.76±0.056	98.87±0.025	99.86±0.036	99.36±0.234	0.028	95.05

ViT, swim transformer, and graph transformer require large and extensive training to capture patterns effectively which make prone to overfitting. However, CNN are better than recent transformer models due to capturing local spatial patterns and hierarchical structures in malware image with less parameters that enable less prone to overfitting. Moreover, the convolutional layers extract fine-grained textures and structural similarities which ensures stable learning across diverse malware families. This efficiency ensures CNN to provide rapid training, less computational complexity, and obtain superior detection accuracy. Hence, the CNN contribute high accuracy with training efficiency by making feature differentiation accurately compared to existing methods.

The t-test is employed to identify whether the difference among experimental outcomes is statistically significant which ensures that the observed enhancements are not because of random chance. A p-value represents the significance with lower values which provides better performance against null hypothesis. Confidence interval is a statistical range which is calculated from sample data within true population parameter with specified probability.

Table 4 represents an evaluation of various hyperparameter tuning. Compared to existing techniques such as RFO, AFO, PFOA, neuroevolution, and Bayesian optimization (BO)-DSPFOA, the proposed DSPFOA obtains a high accuracy of 99.65 and 99.76% using BIG2015 and Malimg dataset by effectively fine-tuning the CNN parameters. But, existing methods such as RFO, AFO, and BO-DSPFOA get trapped in local optima because of the reliance on surrogate models and constraints global search ability. Neuroevolution optimizes both CNN and weights by utilizing evolutionary strategies. However, neuroevolution evaluates many candidate networks over multiple generations which enable highly resource and time-intensive. DSPFOA enhances CNN's feature extraction ability by choosing the best configurations that results in improved detection performance in malware images. Its exploration and exploitation stage makes better convergence, solves the local optima issue, and enhances feature learning effectiveness that improves interpretability, robustness and generalization against different malware families. Therefore, this fine-tuning leads to superior classification accuracy that makes CNN perform more effectively in detecting malware.

Table 4. Performance analysis of different hyperparameter tuning methods on different datasets

Methods	Dataset	Accuracy (%)	Recall (%)	Precision (%)	F1-score (%)
RFO	BIG2015	82.39	87.39	89.30	88.33
AFO		86.49	85.31	92.36	88.69
PFOA		93.29	96.40	94.87	95.62
Neuroevolution		94.12	92.18	93.48	92.82
BO-DSPFOA		95.46	94.03	95.65	94.83
DSPFOA		99.65	98.09	98.97	98.52
RFO	Malimg	85.39	89.19	92.09	90.61
AFO		87.32	90.38	89.43	89.90
PFOA		92.10	92.10	86.92	89.43
Neuroevolution		93.25	91.08	88.49	89.76
BO-DSPFOA		94.85	94.78	90.71	92.70
DSPFOA		99.76	98.87	99.86	99.36

Table 5 provides a performance analysis of k-fold validation for proposed CNN-DSPFOA. Compared to k=3, 7, and 9, k=5 obtains a high accuracy for both dataset due to its balanced trade-off between variance and bias. A lower value like k=3 makes the model more sensitive to outliers and noise, which results in overfitting. Similarly, high values such as k=7 and 9 enhance bias, which makes the decision boundary too smooth and minimizes sensitivity to local patterns. While k=5, the model effectively captures local structures by managing noise influence that ensures better generalization. Therefore, this optimal neighborhood size improves classification performance and interpretability by managing a reliable and stable decision-making process.

Table 5. k-fold validation analysis of proposed CNN-DSPFOA for stability assessment

Dataset	k-fold	Accuracy (%)	Recall (%)	Precision (%)	F1-score (%)
BIG2015	k=3	84.30	82.19	92.13	86.87
	k=5	99.65	98.09	98.97	98.52
	k=7	86.48	84.29	93.81	88.79
	k=9	92.10	89.30	89.02	89.15
Malimg	k=3	87.03	87.31	83.91	85.57
	k=5	99.76	98.87	99.86	99.36
	k=7	89.02	89.30	90.43	89.86
	k=9	92.38	92.10	93.49	92.78

Table 6 depicts the analysis of computational time and memory consumption for different detection methods. The CNN obtains less computational time and memory consumption of 76, 83 s and 87, 98 KB for BIG2015 and Malimg dataset compared to existing methods like ResNet, ViT, and swim transformer due to convolutional layer which minimize the number of parameters. Instead of processing the whole image, CNN employs small kernels that significantly minimize computational complexity. The graphic processing units (GPUs) enhance efficiency and make CNN rapid but less memory efficient.

Table 6. Computational time and memory consumption analysis of different detection and classification methods

Dataset	Methods	Computational time (s)	Memory consumption (KB)
BIG2015	ResNet	102	176
	ViT	89	165
	Swin transformer	82	125
	CNN	76	87
Malimg	ResNet	98	176
	ViT	92	165
	Swin transformer	94	154
	CNN	83	98

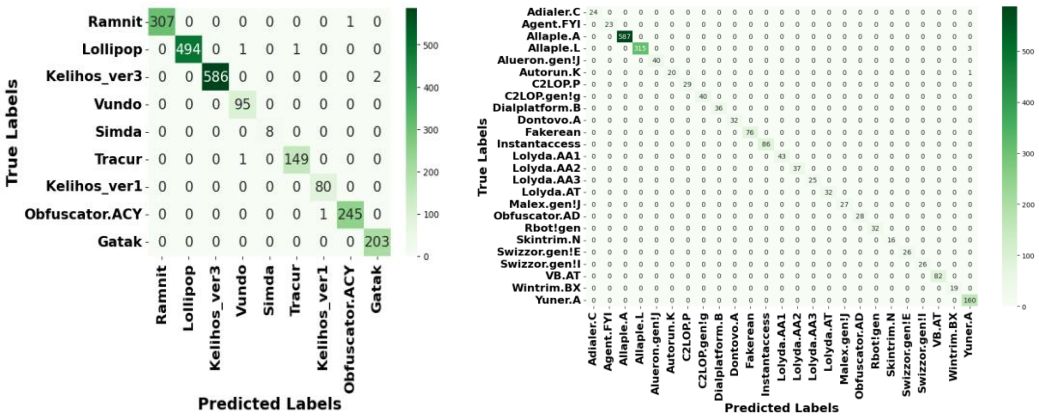
Table 7 demonstrates the evaluation of ablation study for individual component in proposed method. across BIG2015 and Malimg dataset. While compared to individual components such as FOA, PFOA, sin chaotic mapping FOA, Cauchy operator mutation FOA, DSFOA, sin chaotic mapping PFOA, and Cauchy operator mutation PFOA, proposed DSPFOA obtains accuracy of 99.65 and 99.76% due to integrating the benefits of each component. Traditional FOA and PFOA offers global exploration however converges prematurely. By adding sine chaotic mapping improves population diversity and avoids stagnation whereas Cauchy mutation process enhances local search through escaping local optima and PFOA increase stability. By integrating these components, DSPFOA balances exploitation and exploration more effectively that prevents premature convergence and manage different population that results in high accuracy, interpretability, and robustness.

Table 7. Ablation study results analyzing the contribution of individual components in proposed method

Methods	Dataset	Accuracy (%)	Recall (%)	Precision (%)	F1-score (%)
FOA	BIG2015	92.69	90.48	90.48	90.48
PFOA		93.29	96.40	94.87	95.62
Sine chaotic mapping FOA		93.47	96.48	93.59	95.01
Cauchy operator mutation FOA		95.26	96.54	93.96	95.23
DSFOA		95.78	96.59	94.65	95.61
Sine chaotic mapping PFOA		96.78	96.78	94.90	96.78
Cauchy operator mutation PFOA		97.48	97.12	95.36	97.12
DSPFOA		99.65	98.09	98.97	98.52
FOA	Malimg	90.78	84.26	90.45	87.24
PFOA		92.10	92.10	86.92	89.43
Sine chaotic mapping FOA		92.48	92.25	88.45	90.31
Cauchy operator mutation FOA		92.87	92.29	90.74	91.50
DSFOA		92.98	92.45	91.02	91.72
Sine chaotic mapping PFOA		93.60	92.78	91.69	92.78
Cauchy operator mutation PFOA		94.85	93.19	93.26	93.19
DSPFOA		99.76	98.87	99.86	99.36

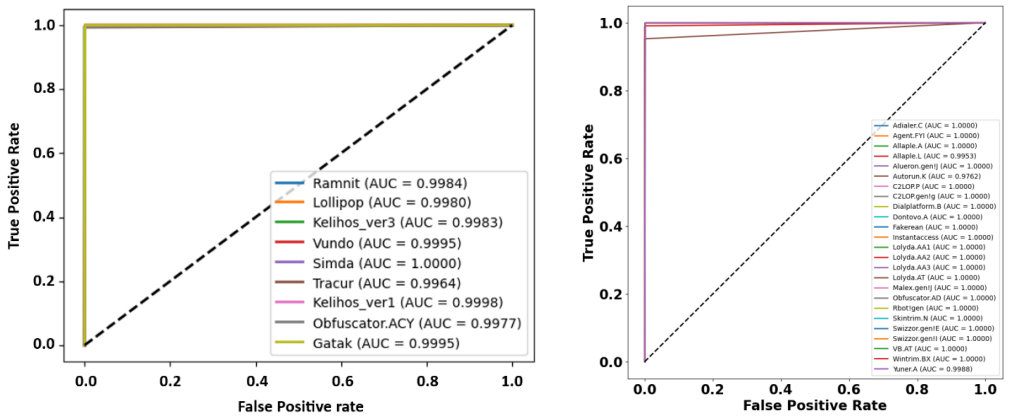
Figure 4 shows confusion matrix for proposed method to classify the different classes: Figure 4(a) for the BIG2015 and Figure 4(b) for the Malimg dataset. In BIG2015, samples are classified accurately with less misclassification. Likewise, model obtains high performance in Malimg dataset across different malware families. The outcomes represents that model efficiently differentiate among different malware families which shows better performance on both datasets.

Figure 5 demonstrates the evaluation of receiver operating characteristics (ROC) curve for proposed method: Figure 5(a) for the BIG2015 and Figure 5(b) for the Malimg dataset. In BIG2015, all malware families obtain high area under the curve (AUC) that represents trade-off among TP and FP. Similarly, model attains better AUC on all classes which shows superior classification performance. Overall, outcomes validate the model reliability for malware classification.



(a) (b)

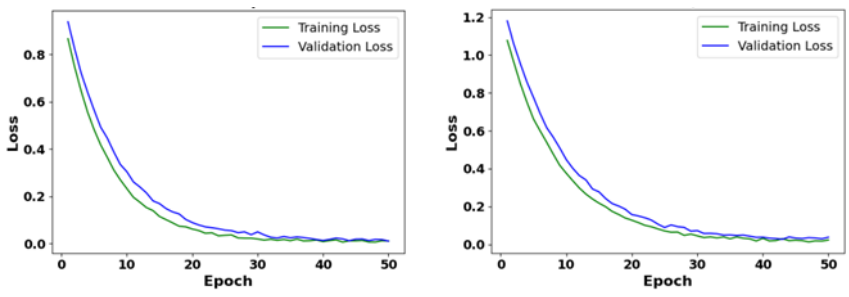
Figure 4. Confusion matrix analysis for proposed method using (a) BIG2015 and (b) Malimg



(a) (b)

Figure 5. ROC curve evaluation of proposed method for malware classification of (a) BIG2015 and (b) Malimg

Figure 6 determines performance analysis of epoch vs loss for proposed method to show convergence behavior: Figure 6(a) for the BIG2015 and Figure 6(b) for the Malimg dataset. During initial epochs, loss minimize sharply which reflects rapid performance for both datasets. As training efficiency progresses, curve flattens which shows stability and less error rates. Moreover, overfitting did not occur due to training and validation loss curve minimize consistently and remain closely without significant gap. This alignment represents that model generalized effectively to unseen data. Also, flattening of both curves at later epochs indicates stable learning which confirms that the model prevents overfitting while maintaining superior accuracy across dataset.



(a) (b)

Figure 6. Performance analysis of epoch vs loss which illustrates convergence behavior of proposed method for (a) BIG2015 and (b) Malimg

Table 8 shows the evaluation of various detection methods on CIC-IDS2017 [26] dataset. This dataset is chosen to evaluate the generalization ability due to containing diverse and intrusion traffic that has multiple attack types that mimic malware behavior. Its richness in network-level features enable highly appropriate for assessing robustness beyond image-based dataset like Malimg and BIG2015. It determines the proposed method effectiveness in capturing discriminative features across unseen and heterogenous malware samples. This confirms that CNN-DSPFOA is scalable and effective for broader cybersecurity applications.

Table 8. Generalization analysis using CIC-IDS2017 dataset for various detection and classification methods

Methods	Accuracy (%)	Recall (%)	Precision (%)	F1-score (%)	t-test from p-value	Confidence interval (%)
ResNet	81.05	79.48	78.05	78.75	0.0035	86.39
ViT	83.49	81.05	80.38	80.71	0.0033	89.78
Swin transformer	85.78	83.69	81.42	82.53	0.0031	89.99
Graph transformer	85.96	84.25	82.49	83.36	0.0031	90.34
CNN	95.25	94.78	95.06	94.91	0.0029	91.48

4.2. Comparative analysis

Table 9 represents the comparative evaluation of proposed CNN-DSPFOA with existing techniques for malware detection and classification. The existing methods, like [15]–[20] are compared with proposed CNN-DSPFOA. Compared to these methods, CNN-DSPFOA obtains a high accuracy of 99.65 and 99.76% for BIG2015 and Malimg datasets by integrating CNN's feature extraction with DSPFOA's effective hyperparameter. CNN captures hierarchical patterns in malware images, whereas DSPFOA fine-tunes the CNN parameter values for optimal performance which enhances feature representation and generalization compared to existing methods.

Table 9. Comparison of the proposed method with existing techniques for detecting and classifying malware

Methods	Dataset	Accuracy (%)	Recall (%)	Precision (%)	F1-score (%)
MalSort [15]	BIG2015	97.85	97.63	97.85	97.85
	Malimg	98.28	98.18	98.19	98.28
ConvNet [16]	BIG2015	96.19	N/A	N/A	N/A
	Malimg	94.22	N/A	N/A	N/A
Hierarchical CloudDNN [17]	BIG2015	98.90	96.88	98.76	97.81
Hybrid model [18]	Malimg	99.06	98.52	98.47	98.49
GRU+CNN+RF [19]	BIG2015	93.74	93.74	93.75	93.69
	Malimg	99.5187	99.52	99.55	99.52
Attention-based cross model CNN [20]	BIG2015	98.72	N/A	N/A	98.00
	Malimg	99.09	N/A	N/A	97.6
Proposed CNN-DSPFOA	BIG2015	99.65	98.09	98.97	98.56
	Malimg	99.76	98.87	99.86	99.76

4.3. Discussion

An advantage of the proposed method and limitation of existing methods are presented in this section in detail. A limitation of existing methods like MalSort [15] struggled with fine-grained malware classification because of the reliance on self-supervised learning that overlooks subtle variations in malware patterns. CNN [16] struggled to detect the malware effectively because of its limited ability to capture fine-grained differences among subtle variants of malware. The hybrid model [18] suffered from feature redundancy because of overlapping extracted features, which resulted in minimized model effectiveness. The proposed CNN-DSPFOA overcomes these existing method limitations. CNN captures spatial patterns in malware images, whereas DSPFOA fine-tunes the CNN values that enhance classification accuracy. The DSPFOA solves local optima, which ensures better convergence and generalization. Therefore, this integration minimizes FP, computational time complexity and enhances robustness, interpretability, against diverse malware families. The deep model extracts highly discriminative features utilizing normalization, it ensures minority classes are learned effectively. Even without using resampling methods, model obtains balanced performance across all malware families. The proposed method advances AI-driven cybersecurity solutions which provides scalable, robust, and accurate malware detection abilities. Overall, the CNN-DSPFOA ensures a more reliable and effective malware detection model compared to existing methods.

5. CONCLUSION

In this research, the CNN-DSPFOA is proposed for malware detection by fine-tuning the CNN's parameter value effectively. CNN capture anomalies and subtle structural patterns across various malware families, which makes it highly efficient at detecting both known and evolving malware. The sine chaotic mapping and Cauchy operator mutation are the di-strategies used in PFOA to optimize the CNN values, which increase robustness and accuracy against evolving malware variants. Min-max scales standardize the image to a fixed range that enhances model stability and consistency by minimizing the sensitivity in varying input values. Through fine-tuning, the proposed CNN-DSPFOA obtains a high accuracy of 99.65 and 99.76% for BIG2015 and Malimg dataset compared to existing methods like MalSort. CNN-DSPFOA can adapt to diverse malware families, which makes it a significant tool for cybersecurity applications. In the future, explainable artificial intelligence (XAI) methods like gradient-weighted class activation mapping (Grad-CAM) and Shapley additive explanations (SHAP) will be incorporated to visualize the malware image regions that influence the CNN's decision and also address adversarial robustness in future work to determine model performance against adversarial samples.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Parvathi Sathenahalli Jayaprakash	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	
Yogeesh Ambalagere Chandrashekaraiah		✓				✓		✓	✓	✓	✓	✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The datasets generated and/or analyzed during the current study are publicly available from the Kaggle repository, including the BIG2015, Malimg, and CIC-IDS2017 datasets, accessible at https://www.kaggle.com/c/malware-classification, https://www.kaggle.com/datasets/manmandes/malimg, and https://www.kaggle.com/datasets/chethuhn/network-intrusion-dataset, respectively.

REFERENCES

[1] K. S. Roy, T. Ahmed, P. B. Udas, M. E. Karim, and S. Majumdar, "MalHyStack: a hybrid stacked ensemble learning framework with feature engineering schemes for obfuscated malware analysis," *Intelligent Systems with Applications*, vol. 20, 2023, doi: 10.1016/j.iswa.2023.200283.

[2] H. Nguyen, F. Di Troia, G. Ishigaki, and M. Stamp, "Generative adversarial networks and image-based malware classification," *Journal of Computer Virology and Hacking Techniques*, vol. 19, no. 4, pp. 579–595, 2023, doi: 10.1007/s11416-023-00465-2.

[3] O. Sharma, A. Sharma, and A. Kalia, "Windows and IoT malware visualization and classification with deep CNN and Xception CNN using Markov images," *Journal of Intelligent Information Systems*, vol. 60, no. 2, pp. 349–375, 2022, doi: 10.1007/s10844-022-00734-4.

[4] M. M. Belal and D. M. Sundaram, "Global-local attention-based butterfly vision transformer for visualization-based malware classification," *IEEE Access*, vol. 11, pp. 69337–69355, 2023, doi: 10.1109/access.2023.3293530.

[5] L. S. Fascí, M. Fisichella, G. Lax, and C. Qian, "Disarming visualization-based approaches in malware detection systems," *Computers & Security*, vol. 126, 2023, doi: 10.1016/j.cose.2022.103062.

[6] F. Ullah, S. Ullah, G. Srivastava, and J. C.-W. Lin, "Droid-MCFG: Android malware detection system using manifest and control flow traces with multi-head temporal convolutional network," *Physical Communication*, vol. 57, 2023, doi: 10.1016/j.phycom.2022.101975.

[7] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "API-MalDetect: automated malware detection framework for windows based on API calls and deep learning techniques," *Journal of Network and Computer Applications*, vol. 218, 2023, doi: 10.1016/j.jnca.2023.103704.

[8] B. G. Doan *et al.*, "Feature-space Bayesian adversarial learning improved malware detector robustness," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 12, pp. 14783–14791, 2023, doi: 10.1609/aaai.v37i12.26727.
[9] J. Park, Ah. Ji, M. Park, M. S. Rahman, and S. E. Oh, "MalCL: leveraging GAN-based generative replay to combat catastrophic forgetting in malware classification," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 1, pp. 658–666, 2025, doi: 10.1609/aaai.v39i1.32047.
[10] M. N. Al-Andoli, K. S. Sim, S. C. Tan, P. Y. Goh, and C. P. Lim, "An ensemble-based parallel deep learning classifier with PSO-BP optimization for malware detection," *IEEE Access*, vol. 11, pp. 76330–76346, 2023, doi: 10.1109/access.2023.3296789.
[11] M. A. Hossain *et al.*, "AI-enabled approach for enhancing obfuscated malware detection: a hybrid ensemble learning with combined feature selection techniques," *International Journal of System Assurance Engineering and Management*, pp. 1–19, 2024, doi: 10.1007/s13198-024-02294-y.
[12] S. Li, J. Wang, Y. Song, S. Wang, and Y. Wang, "A lightweight model for malicious code classification based on structural reparameterisation and large convolutional kernels," *International Journal of Computational Intelligence Systems*, vol. 17, no. 1, 2024, doi: 10.1007/s44196-023-00400-9.
[13] T. Bui, D. Tran, L. G. Nguyen, V. Tong, V. Le, and T. N. Le, "PRAU-GIN: GIN-based android malware classification with traffic refinement and node augmentation," *International Journal of Information Security*, vol. 24, no. 5, 2025, doi: 10.1007/s10207-025-01110-3.
[14] M. U. Rashid *et al.*, "Hybrid android malware detection and classification using deep neural networks," *International Journal of Computational Intelligence Systems*, vol. 18, no. 1, 2025, doi: 10.1007/s44196-025-00783-x.
[15] F. Wang *et al.*, "MalSort: lightweight and efficient image-based malware classification using masked self-supervised framework with swin transformer," *Journal of Information Security and Applications*, vol. 83, 2024, doi: 10.1016/j.jisa.2024.103784.
[16] Y. Liu, H. Fan, J. Zhao, J. Zhang, and X. Yin, "Efficient and generalized image-based CNN algorithm for multi-class malware detection," *IEEE Access*, vol. 12, pp. 104317–104332, 2024, doi: 10.1109/access.2024.3435362.
[17] M. R. B. Mosleh and S. Sharifian, "An efficient cloud-integrated distributed deep neural network framework for IoT malware classification," *Future Generation Computer Systems*, vol. 157, pp. 603–617, 2024, doi: 10.1016/j.future.2024.03.051.
[18] K. Shaukat, S. Luo, and V. Varadharajan, "A novel deep learning-based approach for malware detection," *Engineering Applications of Artificial Intelligence*, vol. 122, 2023, doi: 10.1016/j.engappai.2023.106030.
[19] S. Singh, D. Krishnan, V. Vazirani, V. Ravi, and S. A. Alsuhibany, "Deep hybrid approach with sequential feature extraction and classification for robust malware detection," *Egyptian Informatics Journal*, vol. 27, 2024, doi: 10.1016/j.eij.2024.100539.
[20] J. Kim, J.-Y. Paik, and E.-S. Cho, "Attention-based cross-modal CNN using non-disassembled files for malware classification," *IEEE Access*, vol. 11, pp. 22889–22903, 2023, doi: 10.1109/access.2023.3253770.
[21] Microsoft, "Microsoft malware classification challenge (BIG 2015)," *Kaggle*. Accessed: Feb. 18, 2025. [Online]. Available: https://www.kaggle.com/c/malware-classification
[22] J. Xu, "Malimg dataset," *Kaggle.* Accessed: Feb. 18, 2025. [Online]. Available: https://www.kaggle.com/datasets/manmandes/malimg
[23] G. S. Nijaguna, J. A. Babu, B. D. Parameshachari, R. P. de Prado, and J. Frnda, "Quantum fruit fly algorithm and ResNet50-VGG16 for medical diagnosis," *Applied Soft Computing*, vol. 136, 2023, doi: 10.1016/j.asoc.2023.110055.
[24] G. Bompem and D. Pandluri, "Batch normalization based convolutional neural network for segmentation and classification of brain tumor MRI images," *International Journal of Intelligent Engineering and Systems*, vol. 17, no. 2, pp. 39–49, 2024, doi: 10.22266/ijies2024.0430.04.
[25] A. Ghiaskar, A. Amiri, and S. Mirjalili, "Polar fox optimization algorithm: a novel meta-heuristic algorithm," *Neural Computing and Applications*, vol. 36, no. 33, pp. 20983–21022, 2024, doi: 10.1007/s00521-024-10346-4.
[26] H. N. Chethan, "Network intrusion dataset (CIC-IDS- 2017)," *Kaggle*. Accessed: Sep. 23, 2025. [Online]. Available: https://www.kaggle.com/datasets/chethuhn/network-intrusion-dataset

BIOGRAPHIES OF AUTHORS

Parvathi Sathenahalli Jayaprakash has completed her B.E. and M.Tech. from Visvesvaraya Technological University (VTU), Belagavi, Karnataka, India. She is currently working as an assistant professor in the Department of Information Science and Engineering at JSS Science and Technology University, Mysuru, Karnataka, India. She is pursuing her Ph.D. (part-time) at the Government Engineering College, Chamarajanagar, affiliated to VTU. Her research interests include deep learning, cyber security, and machine learning. She can be contacted at email: sjparvathi@jssstuniv.in.

Yogeesh Ambalagere Chandrashekaraiah has completed B.E., M.Tech., and Ph.D. from Visvesvaraya Technological University Belagavi, Karnataka, India. Currently working as an associate professor in Computer Science and Engineering, Government Engineering College, Chamarajnagar, Karnataka, India. His area of interest is wireless sensor network, IoT, and machine learning. He can be contacted at email: yogeesh13@gmail.com.