Journal of Information Technology Management
ISSN-OnLine: 2980-7972
University of Tehran

RES2NET-MAXOUT: Image-Based Multi-Class Malware Classification with Deep Learning Models

G.Joel Sunny Deol*

*Corresponding author, Prof. and HOD, Department of Computer and Engineering, Kallam HaranadhaReddy Institute of Technology (KHIT), Andhra Pradesh 522019, India. E-mail: sunnydeoldrgjoel@gmail.com

Kanakam Siva Rama Prasad

Associate Prof. and HOD, Department of Information Technology, Kallam Haranadhareddy Institute of Technology, Andhra Pradesh 522019, India. E-mail: drksivaramaprasad@gmail.com

Alahari Hanumat Prasad

Prof. and HOD, Department of Computer and Engineering- Artificial intelligence & Data Science, Kallam Haranadhareddy Institute of Technology, Andhra Pradesh 522019, India. E-mail: drahanumatprasad@gmail.com

M. Chennakesava Rao

Associate prof., Computer and Engineering- Artificial intelligence & Machine learning, Kallam HaranadhaReddy Institute of Technology (KHIT) (AUTONOMOUS), Chowdavaram, Andhra Pradesh, India. E-mail: mchennakesavarao.khit@gmail.com

Medida Alekhya

Assistant prof., Computer and Engineering- Artificial intelligence & Machine learning, Kallam HaranadhaReddy Institute of Technology (KHIT) (AUTONOMOUS), Chowdavaram, Andhra Pradesh, India. E-mail: medidaalekhya.khit@gmail.com

T. Alekhya

Assistant prof., Computer and Engineering- Artificial intelligence & Machine learning, Kallam HaranadhaReddy Institute of Technology (KHIT) (AUTONOMOUS), Chowdavaram, Andhra Pradesh, India. E-mail: talekhya.khit@gmail.com

Journal of Information Technology Management, 2026, Vol. 18, Issue 3, pp. 1-24
Published by the University of Tehran, College of Management
doi: https://doi.org/10.22059/jitm.2026.388431.3967
Article Type: Research Paper

Received: January 25, 2026
Received in revised form: March 13, 2026
Accepted: May 04, 2026
Published online: July 01, 2026

Abstract

The dynamic nature of the Internet has resulted in an increase in network security threats, primarily caused by the growing incidence of malware and the continuous emergence of its many forms. Conventional methods for detecting malware involve complex feature engineering, which considerably reduces detection effectiveness. Furthermore, machine learning algorithms, which were once a mainstay in the fight against cyber threats, are no longer able to accurately detect every new and complex malware variation. In contrast, deep learning approaches have the potential to be transformative. Their inherent flexibility and capacity for pattern recognition present a viable solution to the ever changing challenge of detecting a wide range of malware variations more reliably and effectively. For this reason, a unique deep learning based architecture is developed in this study to categorize malware types and protect networks. Three components comprise the most significant parts of this work: feature extraction, classification, and malware visualization. The suggested method first converts malware binary data into two dimensional images. Secondly, we introduce an automatic feature extractor that utilizes the Res2Net model to extract shared patterns among family members from the visualized images. Lastly, we propose a Deep Maxout Network designed to categorize malware using the extracted attributes. Moreover, data augmentation based on Conditional Generative Adversarial Networks (CGAN) addresses the issue of imbalanced data. The proposed technique is evaluated using the MaleVis, Microsoft Malware Classification Challenge (MMCC), and Drebin datasets. The experimental findings demonstrate that the proposed method can successfully and accurately classify malware compared to current state of the art techniques.

Keywords: Deep learning, Res2Net, CWGAN, Malware, Deep Maxout Network

Introduction

The Internet and contemporary computer technologies have enhanced people's quality of life and increased convenience. Many tasks can now be performed online, including communicating with others, conducting monetary transactions, monitoring physical changes, and much more. These developments have also encouraged cybercriminals to commit crimes online rather than offline (Sharma et al., 2024; Kumar et al., 2024; Maray et al., 2023). According to recent scientific and industry reports, cyberattacks have caused trillions of dollars in losses to the global economy. Cybercriminals typically use malware to initiate cyberattacks. Malware is software that exploits a victim's machine to perform unknown or suspicious tasks. It commonly appears in the form of worms, Trojan horses, rootkits, botnets, and spyware (Dhalaria et al., 2024; Tang et al., 2023; Alzubi et al., 2023). Malware can be used for illegal activities such as stealing data, exploiting hardware resources, or causing operational disruptions for monetary or reputational gain.

Malware is evolving so rapidly that classifying malware families has become essential. Less than 2% of the code typically varies between versions of known malware, yet these minor modifications account for a significant proportion of newly discovered variants (Das et al., 2024; Jiang et al., 2023; Son et al., 2022). As a foundation for malware classification, this suggests that most samples belonging to the same family share functional similarities. In this context, cybersecurity increasingly depends on the ability to quickly and accurately categorize the vast number of emerging malware variants.

Static and dynamic analysis are the primary methods used in classical malware classification. Static analysis techniques operate by disassembling malicious binary data and directly retrieving relevant information, such as hash values, processes, and string lists (Alamro et al., 2023; Zhan et al., 2023; Chaganti et al., 2022). This eliminates the need to execute malware binaries. These techniques are known for their rapid processing times, ease of use, and relatively high accuracy. However, their analytical limitations stem from their superficial nature, making them vulnerable to obfuscation strategies such as code distortion and hindering their ability to identify or categorize unfamiliar malware (Eslavath et al., 2024; Aurangzeb et al., 2023; Tekerek et al., 2022).

In contrast, dynamic analysis techniques operate in virtual environments and are not affected by obfuscation mechanisms. They enable the detection of novel malware variants by monitoring the real-time behavior and modifications of malware binary samples. Nevertheless, dynamic analysis is often highly complex and time-consuming.

Machine learning approaches have recently shown considerable promise in malware classification (Yadav et al., 2023; Demirkıran et al., 2022; Othman et al., 2024). When training data is limited, traditional machine learning techniques may outperform deep learning models. However, adapting classical machine learning algorithms to complex and evolving environments remains challenging due to their restricted flexibility, reliance on extensive feature engineering, and limited capacity to learn complex attribute relationships. As the volume and complexity of malicious programs continue to grow exponentially, traditional machine learning approaches are increasingly unable to meet the requirements for effective malware detection and classification in modern cybersecurity contexts (Liu et al., 2022; Parihar et al., 2022).

Deep learning technologies, by contrast, offer notable advantages for processing large-scale data. They reduce the need for manual feature engineering by automatically extracting high-level representations and can effectively handle unstructured data. The Q deep network algorithm is widely used due to its ability to learn optimal policies in complex environments. As reinforcement learning models are often too complex to analyze, numerical experiments and simulations are used to analyze different pricing strategies (Niknami & Ajhondzadeh Noughabi, 2024).

Motivated by these limitations, the present study proposes a framework for malware classification that integrates deep learning networks with image-based representations. The framework effectively addresses challenges related to malware detection and variant recognition. The proposed architecture employs the powerful representational capabilities of Res2Net networks to extract intricate features from malware images. These distinctive characteristics are then used to improve family-level malware classification through an optimized Deep Maxout Network. The adaptive Maxout units within this network adjust their behavior based on input characteristics, enhancing robustness against diverse and evolving malware threats. Furthermore, the imbalance present in the dataset is addressed through Conditional Generative Adversarial Networks (CGANs), which augment minority classes to improve training stability and reduce overfitting.

The main contributions of this study are summarized as follows:

- The study emphasizes the value of CGAN-based data augmentation methods in resolving issues with imbalanced data. As a result, the training set became more representative, improving the method's capacity to capture a wide range of properties and variations. The increased dataset improves the model's overall performance by strengthening its stability and ability to generalize to previously undiscovered cases.

- Using Res2Net-based feature extraction, more intricate virus patterns are learned. More different patterns in the data are captured, which helps to enhance the categorization process.

- A fresh approach to malware categorization is put forth, utilizing the deep maxout network to lessen the need on feature engineering methods and domain knowledge. It also offers an easy-to-use deployment process and a high categorization accuracy. Furthermore, the hyperparameter optimization using the Coot optimization algorithm (COA) improves its performance.

- Several tests are conducted using three malware datasets to compare the effectiveness of the proposed approach with other pertinent approaches.

The remainder of the document is structured as follows: The relevant research on a few contemporary and well-liked malware categorization methods is covered in Section 2. The third section provides a detailed illustration of the suggested model. Section 4 assesses how well the suggested strategy performs. Finally, Section 5 provides a summary of our work.

Literature Review

Alomari et al. (2023) created a malware identification framework using feature selection techniques and deep learning. To identify malware and distinguish it from innocuous activity,

two distinct malware datasets are utilized. After preprocessing, the datasets are subjected to correlation-based feature selection to select more distinct features. Afterwards, LSTM-based deep learning models were trained on these different feature-selected datasets and categorized the malware samples. After training, this approach was assessed using a variety of performance metrics, such as F1-score, recall, precision, and accuracy.

El-Ghamry et al. (2023) presented an image-based machine learning method for identifying IoT malware. To improve malware detection results, their approach combined an ant colony optimizer (ACO)-based feature selection technique to reduce the number of features while boosting the efficiency of the SVM classifier. Furthermore, SVM parameters were optimized across several kernel functions using the Particle Swarm Optimization (PSO) algorithm. Their investigations, using a publicly accessible dataset, show that the suggested method performs better than others, obtaining 95.56% accuracy, 95.26% F1-score, 96.43% recall, and 94.12% precision.

Ravi and Alazab (2023) proposed an end-to-end, lightweight, multi-headed attention deep learning method for malware image categorization. This approach focuses on learning the crucial areas of the malware and improves the visual representation power of convolutional features. The suggested method converts malware into grayscale images, which are then fed into deep learning for automated feature extraction and categorization. Learning pertinent characteristics for efficient feature representation and identifying smaller portions of malware regions in the overall image becomes more accessible by integrating attention into the CNN.

Kim et al. (2023) focused on the complex problem of malware family categorization. This method avoids disassembly and presents a novel convolutional neural network (CNN)-based model for categorizing malware files that are not deconstructed, particularly binary files. Two separate modalities, “structural entropies” and “malware images,” both generated and retrieved from binary files, were integrated into the model. These modalities offer a complementary viewpoint by capturing varying granularities of bytes and blocks. The model modifies the expressive limits inherent in each modality by using a cross-modal attention mechanism to efficiently synergize information from these modalities. Experiments on three well-known datasets—the BIG2015, Malimg, and BODMAS datasets—validate the resilience and effectiveness of the suggested approach and demonstrate its performance.

Zou et al. (2024) proposed a new image-based malware classification method based on a capsule network classification model. This approach convolves malware images during the first feature extraction stage to accomplish multi-level feature extraction. It then combines the extracted characteristics from different levels using dynamic convolution. To minimize data loss and enhance training stability, they incorporate a set of balancing factors in the dynamic routing of primary capsules. In the end, they obtain the final outputs and description vectors

of high-level capsules at various stages. Experiments were conducted using the VIRUS-MNIST, Malimg, and BIG2015 datasets.

Nikam and Deshmukh (2024) recommended a hybrid feature-based malware classification approach. Information gain, chi-square, and feature importance strategies were employed to select features. This method investigates the impact of feature selection on classifier performance. First, each technique was used to choose the top 20 significant features, and classifier performance was evaluated. Subsequently, all 60 features were combined, and features common to more than one approach were designated as hybrid features. It was found that 11 out of 60 features were hybrid features. The effectiveness of classifiers such as XGBoost, decision trees, KNN, SVM, and linear regression was reexamined using these 11 hybrid features. Compared to performance using individual feature sets, the results show that hybrid features enhance the performance of all classifiers. XGBoost was the most accurate classifier among the five tested.

Chen and Cao (2023) presented an image-based malware categorization framework that combines the capabilities of three traditional neural networks (MobileNetV2, MobileNetV1, and ResNet50) using ensemble learning to directly extract useful features from malware images. To reduce data imbalance, the authors pretrained the networks through transfer learning. After feature extraction, principal component analysis was applied to reduce dimensionality, and support vector machines were used for malware classification. The Malimg dataset was employed to evaluate the effectiveness of this method under different performance metrics.

Methodology

Our suggested deep learning-based malware categorization framework is shown in this section. For malware categorization, this system offers a hybrid deep learning framework. The approach of the proposed system, represented in Figure 2, contains four essential components. First, several comprehensive datasets are used to gather malware data. These malicious data are transformed into grayscale malware images for accurate malware analysis. Second, a CGAN-based approach is used to augment the malware images to obtain a balanced dataset. Thirdly, an efficient Res2Net-based network extracts high- and low-level features from the malware images. Lastly, an optimized Deep maxout network classifies the malware family. Furthermore, to improve the classifier's performance, the hyperparameters are optimized by the Coot optimization algorithm (COA). It guarantees that the model generalizes well to new data and can speed up the process of training convergence. This is important for malware classification because the field is constantly changing.

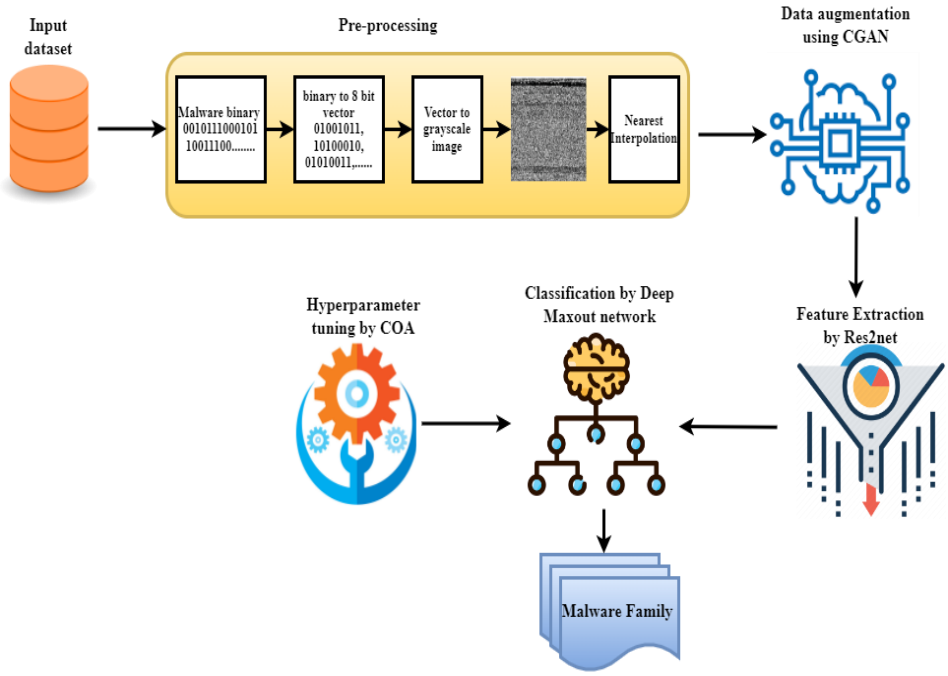

Figure 1. System architecture

Pre-processing

Pre-processing has to be completed before model creation. The MaleVis, Drebin, and MMCC datasets are utilized in the present research to evaluate performance. Malware images can be found in the MaleVis collection, and binary files can be found in the Drebin and MMCC databases. Thus, we converted the binary files into malicious images. In general, binary code can be translated into visuals in several ways. Initially, a vector of 8-bit unsigned integers has been employed to read the malware binary file. Next, equation (1) transforms the binary value of each component A = (a7+a6+a5+a4+a3+a2+a1+a0) to its corresponding decimal value.

$$A = (a_0 * 2^0 + a_1 * 2^1 + a_2 * 2^2 + a_3 * 2^3 + a_4 * 2^4 + a_5 * 2^5 + a_6 * 2^6 + a_7 * 2^7) \quad (1)$$

The decimal vector produced at the end is transformed into a 2D matrix and read as a grayscale picture. The binary files' initial sizes were different. As a result, 256*256 dimensions are applied to every image. While maintaining the integrity of the original data, the nearest interpolation method is employed to resample the pictures during this procedure. It chooses the pixel value nearest to the neighboring locations of the supplied interpolation point. This technique collects the equivalent pixel in the original input figure for each Pixel in the final image. Regarding ease of use, calculation time, and the capacity to preserve initial values in an immutable environment, the nearest interpolation methodology offers advantages over other interpolation techniques, such as bicubic and bilinear interpolation.

Data Augmentation using CGAN

In order to address the imbalanced problem, this section employs the Conditional Generative Adversarial Network (CGAN) technique. It enhances data sets that closely resemble the original information. While labeling the generated samples is restricted, CGAN uses the same training procedure as GAN. This structure consists of two components. They are networks of generators (GRR) and discriminators (DRS). In the process of creating artificial information, these two networks compete.

$$L(DR_S(O),1) = log(\,DR_S(O)) \tag{2}$$

$$L(DR_S(GR_R(b_r)),0) = log(\,1 - DR_S(GR_R(b_r))) \tag{3}$$

In the above equations, br signifies an input set of random variables from the GRR network, and DRS (O) refers to the probability that the original data 'O′ is predicted by the DRS network. The aforementioned equations can be integrated and maximized to determine the discriminator's total loss function because the discriminator's purpose is to discern between authentic and fraudulent data accurately.

$$L^{(DR_S)} = max[log(\,DR_S(O)) + log(\,1 - DR_S(GR_R(b_r)))] \tag{4}$$

Unlike the discriminator, the generator's loss function is shown by,

$$L^{(GR_R)} = min[log(\,DR_S(O)) + log(\,1 - DR_S(GR_R(b_r)))] \tag{5}$$

Equation (6) defines the value function V(GRR, DRS), considering the complete dataset.

$$\min_{GR_R}\max_{DR_S} V(GR_R, DR_S) = \min_{GR_R}\max_{DR_S}(E_{a\sim P_{data(a)}}[log\,D\,R_S(O\ \ )] + (E_{b_r\sim P_{data(b_r)}}[log(\,1 - DR_S(GR_R(b_r)))] \tag{6}$$

Here, $E_{b_r\sim P_{data(b_r)}}$stands for the data distribution received from the generator, $E_{a\sim P_{data(a)}}$the predicted outcome of the original data instances, and Pdata(a) reflects the actual data distribution. $E_{b_r\sim P_{data(b_r)}}$indicates the predicted return of complete random inputs (false data).

The generator network up-samples the given input vectors of features utilizing transposed convolutional layers. Several transposed convolutional layers are employed, with channel counts ranging from 64 to 512. The blocks at each level show the dimensions of the input feature vector. Finally, the structure of the samples produced by this model is similar to that of the original data.

Res2Net based feature extraction

In this subsection, we extract features from the malware images using Res2Net based network. Res2Net performs better in terms of generalization than ResNet. The residual cell structure of the model has hierarchical tiny residual blocks, which broadens every layer's effective perceptual field and improves the network's overall feature extraction performance. A collection of 3 x 3-sized convolutional kernels is used by each Res2Net Block; these kernels are hierarchically coupled to break the feature map into numerous channels. Due to this, Res2Net can capture and process data at many scales because of its hierarchical feature integration, which makes it possible for the network to deal with various characteristics and forms in the data.

Res2Net differs from traditional residual networks in that it divides the input features evenly into 's' groups by the number of channels xi, where i ∈ {1, 2,..., s}. For every xi (apart from x1), there is a corresponding 3 × 3 convolution, represented by ki; the output of ki is represented by yi, which can be expressed by the following equation.

$$y_i = \begin{cases} x_i\,, & \mathrm{i} = 1 \\ K_i(x_i)\,, & \mathrm{i} = 2 \\ K_i(x_i + y_{i-1})\,, & 2 < i \leq s \end{cases} \tag{7}$$

The previous feature subset, which K_i (.) gets the output of the feature subsety_{i-1}, corresponds to a series of distinct convolution operations performed by K_i (.) and$K_{\text{i-1}}$. The perceptual field of each convolutional layer is increased by the construction of interconnecting channels in groups within the residual blocks. It enables the model to enhance the receptive field of every network layer and retrieve finer-grained multiscale characteristics. The final feature map is created by combining all outputsy_i into a single output and feeding it into the output layer following a 1x1 convolution operation.

Increasing the amount of convolutional layers in the Res2Net structure results in an increase in both the network load and the learning time cost. An enhanced grouped convolution module was added after the 1×1 convolution to solve this issue by lowering the model's parameter count and number of operations. To improve the grouped convolutional module's capacity for feature extraction, the inverse bottleneck structure was taken from the MobileNet V2 model. It lowers the complexity of the model and enables the extraction of additional feature information.

Deep Maxout network-based classification

After the features from the malware images are extracted, they are forwarded to the Deep Maxout Network for further malware classification. One kind of deep neural network is the deep max-out network, which has multiple max-out networks connected one after the other in a multilayer structure. The input, embedding, dropout, convolution, maxpooling, dense layer

with activation, maxout layer with maxout function, and output layers are the many layers that make up this network structure. Equation (8) represents this network's maxout unit:

$$Q(M) = \max_{z\in[1,\eta]} R_{yz}, \quad (8)$$

Here, the Maxout unit's index is denoted by yz, η denotes the feature map, bias factor is represented by γ, weight denoted by λ and the input is denoted by M.

There are two Maxout units in the convolutional Maxout layer. One convolutional layer and a channel-wise Max-pooling layer are present in every Maxout unit. Maximizing each of the k separate feature maps within a network can create the Maxout feature mapping. To reach the Maxout operator when using the Dropout method, the input components must first be multiplied by the Dropout mask and then by the weights. Within this framework, maxout units can be understood as giving a piecewise linear behavior that approximates complex relationships. Maxout neurons choose the maximal activity from a range of options, which allows them to capture complex correlations in the data. The relationship between hidden units and the purpose of each hidden unit can be learned through this method. In the network's last layer, the softmax function is performed to construct the output class of the model based on the probabilities derived from the feature maps. The following subsection explains how COA is used to tune the hyperparameters of this network to enhance the performance of the deep maxout model.

Hyperparamter tuning using COA

Coots are small water birds of the rail family Rallidae. This Coot optimization algorithm (COA) is built using the water surface behavior of Coots. On the water's surface, coot groups exhibit four distinct patterns of movement: random movement, chain movement, positioning in response to the leader, and leader movement. These four movements' behaviors are used to create the COOT algorithm, and their mathematical model is explained as follows,

Initially, use Equation (9) to create the starting population.

$$\text{Cootpos}(i) = \text{rand}(1,d) * (ub - lb) + lb \quad (9)$$

Here, d is the number of choice variables, lb, and ub are the lower and upper bounds of the search space, and Cootpos(i) is the location of the ith coot.

'NL' group leaders and 'Ncoot' number of Coots are randomly chosen during population initiation. The four-movement behaviors outlined below are then used to update the positions of the Coots and leaders.

Random Movement:

In order to carry out this movement, random positions are taken into account based on equation (10).

$$Q = \text{rand}(1, d) .* (ub - lb) + lb \quad (10)$$

The coot moves in this way to investigate different areas of the search arena. If the algorithm gets trapped in the local optima, this motion will help it escape. Equation (11) is used to find the new position of the coots.

$$\text{CootPos}_{new}(i) = \text{CootPos}(i) + A \times R_2 \times (Q - \text{CootPos}(i)) \quad (11)$$

Here R_2 is an arbitrary number within the range [0, 1]. Equation (12) is utilized for the computation of A.

$$A = 1 - L \times \left(\frac{1}{Max_{iter}}\right) \quad (12)$$

Here, Maxiter represents the maximum iteration, and L denotes the current iteration.

Chain Movement:

The average distance among two coots can be used to create the chain movement. Equation (13) is used to calculate the coot's new location.

$$\text{CootPos}(i) = 0.5 \times (\text{CootPos}(i-1) + \text{CootPos}(i)) \quad (13)$$

Adjusting the position based on the group leader

Only a few coots lead their flock; the others must adjust their positions at the command of the leader. After accounting for the positions of the leaders, cocks will reposition themselves based on their average placements. We employ equation (14) to put this movement into practice.

$$K = 1 + (i \; MOD \; NL) \quad (14)$$

Here, I denote the current coot's index number, the leader's index number denoted by K, and the leader number denoted by NL.

The location of coot (i) needs to be updated based on the leader's K. Equation (15) computes the forthcoming position of the coot depending on the chosen leader.

$$Cootpos(i) = LeaderPos(k) + 2 \times R_1 \times \cos(2R\pi) \times (LeaderPos(k) - \text{CootPos}(i)) \quad (15)$$

Here, R indicates the chosen leader in the intervals [−1,1], π is 3.14, R is the random number among [0, 1], and $LeaderPos(k)$ denotes the selected leader.

Leader movement

Equation (16) requires the leader to adjust its position to obtain the best solution.

$$\begin{cases} B * R_3 * cos(2\pi R) * (g_{\text{best}} - Leaderpos(i)) + g_{best} & R_4 < 0.5 \\ B * R_3 * cos(2\pi R) * (g_{\text{best}} - Leaderpos(i)) - g_{best} & R_4 \geq 0.5 \end{cases} \quad (16)$$

Here, the random number among -1 and 1 is denoted by R, the global best location is denoted byg_{best}, the random number among 1 and 0 is denoted by R3 and R4, and B is computed using the following equation.

$$B = 2 - L \times \frac{1}{\text{Max}_{\text{iter}}} \quad (17)$$

The procedures mentioned above are repeated until the ideal result is reached. Finally, the Deep Maxout network's hyperparameters, including learning rate, epoch, batch size, weight decay rate, and momentum, are initialized using the optimal COA technique result.

Results and Discussion

This section provides a detailed explanation of the proposed framework's implementation, experimental results, and evaluation. Our research used a 32 GB RAM system with an Intel Core i9 CPU running at 4.8 GHz on Windows. Using the Python programming language, the suggested architecture was implemented.

Performance metrics

To demonstrate the effectiveness of the suggested techniques, multiple assessment measures were employed. Accuracy, precision, sensitivity, specificity, and f-score are these measurements. The performance metrics are computed in the following manner:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (18)$$

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

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

$$Specificity = \frac{TN}{TN+FP} \quad (21)$$

$$F1 - score = 2 * (precision * recall)/(precision + recall) \quad (22)$$

Here, false positive, false negative, true negative, and true positive are denoted by FP, FN, TN, and TN, respectively. The previously mentioned performance metrics are the foundation for interpreting the suggested model's performance.

Dataset Description

MaleVis dataset: The Malware Evaluation with Vision (MaleVis) dataset is the first dataset used in this experiment. There are 5,126 malware instances for testing and 9,100 for training, covering 25 different malware classes. There are 350 training samples and several testing samples in every class. It contains 26 malware samples, and some of them are Fasong, Expiro-H, Elex, BrowseFox, Androm, and Dinwod!rfn. This dataset was converted into images using malware binary files using Sultanik's bin2png program.

MMCC dataset: This dataset was acquired by Microsoft and released by Kaggle for a 2015 competition—twenty-two thousand seven hundred forty-one malware samples from nine distinct classes— Gatak, Obfuscator.ACY, Simda, Vundo, Tracur, Kelihosver3, Kelihos_ver1, Lollipop, and Ramanit—are included in it. The distribution of malware samples across classes is not consistent. Every malware sample consists of two files with the extensions ".byte" and ".asm." In this case, the ".bytes" file contains the unprocessed hexadecimal version of the binary data.

Drebin dataset: This dataset contains the collection of Android malware collected among August 2010 and October 2012. There are 131,611 programs for both safe and risky software in it. There are 12000 safe apps and 5560 malicious apps in the final dataset. Particularly, the collection consists of 2,810 apps from Russian markets, 19,545 apps from different Chinese markets, 96,150 apps from Google Play, and 13,106 apps from other sources such as Android sites, malware forums, and security blogs. Moreover, the dataset contains all samples from the Android Malware Genome Project.

Network training and testing

In this work, three malware datasets are utilized. These datasets are randomly split at 20% and 80% for testing and training datasets, and the number of epochs was set to 100, the batch size was 32, the momentum was 0.8, the weight decay rate at 1e-06, and the 0.001 learning rate was optimized using the COA method. Figures 2, 3, and 4 display the accuracy and loss graphs for the three datasets. Based on the figures, we can observe that accuracy increases

when the number of epochs increases and that training eventually approaches a plateau. Between 99% and 100% is the range of training accuracy when tending to stabilize. The loss value swiftly converges, steadily drops, and stabilizes as the number of epochs rises, as seen in the loss graph. The CGAN-based data augmentation and COA-based hyperparameter optimization are responsible for this stability. It lessens overfitting and data imbalance.

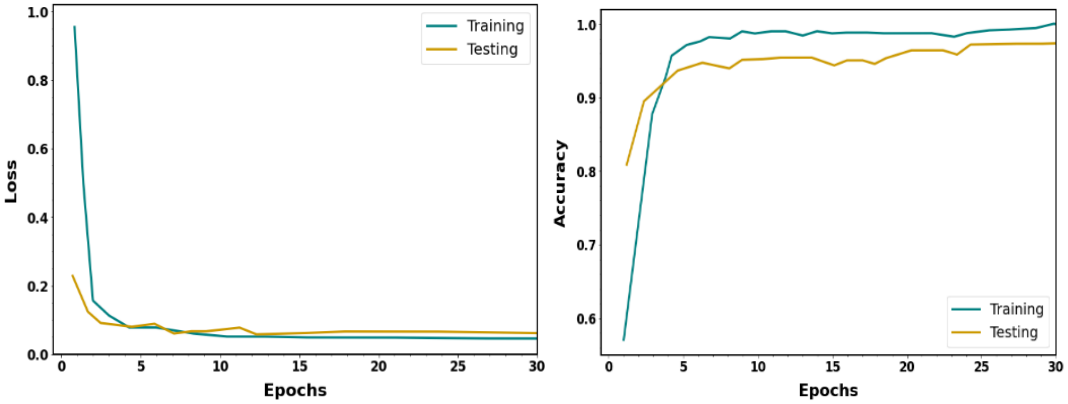

Figure 2. Accuracy and loss graph on the MaleVis dataset

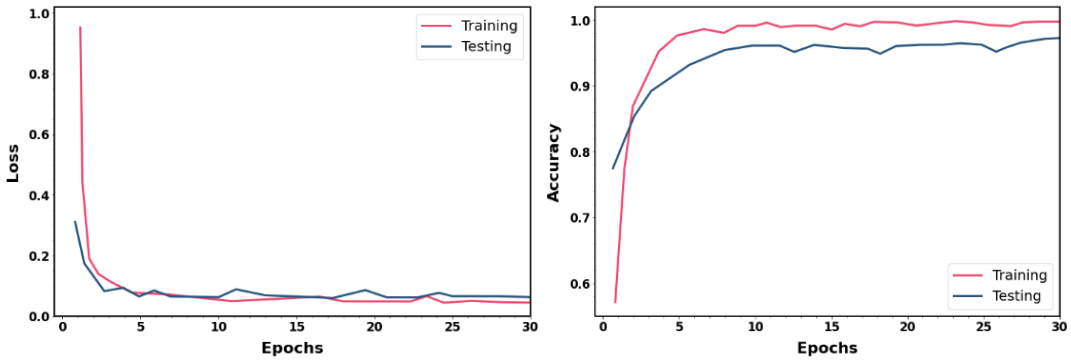

Figure 3. Accuracy and loss graph on the MMCC dataset

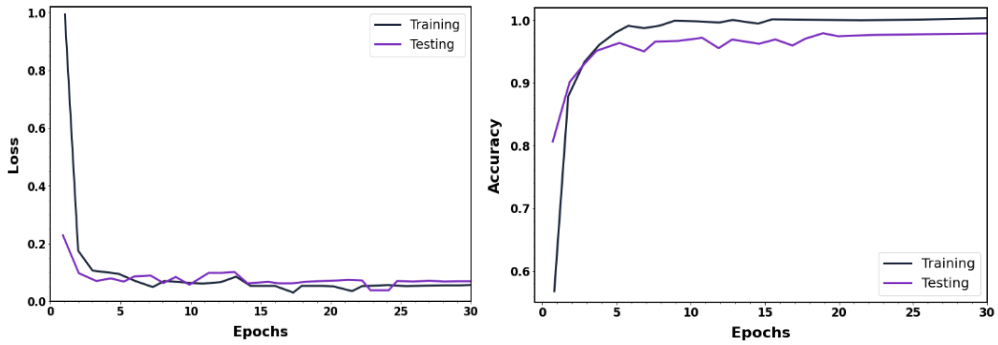

Figure 4. Accuracy and loss graph on the drebin dataset

Performance evaluation on MaleVis dataset

Numerous research works have suggested approaches for categorizing malware families. This part compares our suggested modal with the malware classification models from earlier

research. The models' respective performances are examined in Table 1, and the MaleVis dataset was used to assess each model. As demonstrated in Table 1, the suggested model outperforms the others thanks to utilizing the derived features from the Res2Net-based deep learning technique. Most notably, our methodology proved more resilient to data imbalances because of CGAN-based data augmentation. Our suggested framework attained 99.32% accuracy, 99.27% precision, 99.25% sensitivity, 99.29% specificity, and 99.26% f1-score because of the integration of this multiple hybrid technique.

Table 1. Comparative analysis on the MaleVis dataset

Techniques	accuracy	precision	sensitivity	specificity	F1-score
DenseNet (Hemalatha et al., 2021)	98.21	98.56	97.74	-	98.15
Random forest-based voting (Atitallah et al., 2022)	98.68	98.74	98.67	98.79	98.70
DCNN (Falana et al., 2022)	96.7	94.24	93.4	-	93.4
DNN (Aslan et al., 2021)	96.6	-	97.1	94.9	94.5
SFTA-GDA (Ahmed et al., 2022)	98	-	-	-	-
Proposed	99.32	99.27	99.25	99.29	99.26

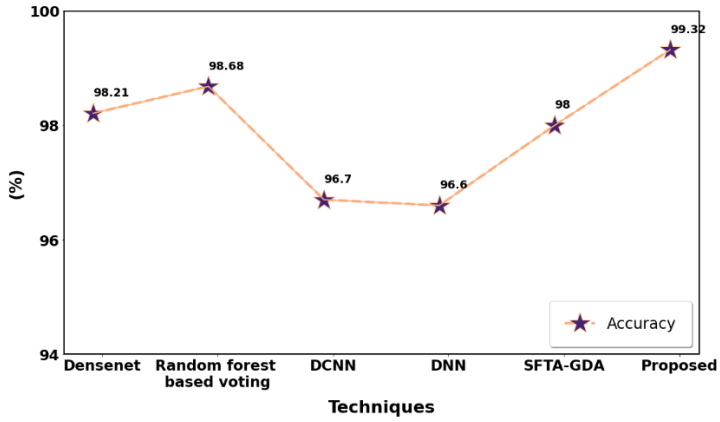

Figure 5. Accuracy comparison on the MaleVis dataset

We realized from the table that the Deep Neural Network's (DNN) performance is marginally worse than other methods. It may have happened due to unbalanced datasets, which can cause difficulties for DNNs and leave some malware classifications underrepresented. When trained on unbalanced data, models may behave biasedly and have trouble correctly classifying minority classes. However, our study uses a CGAN-based technique to handle this problem and get superior outcomes. Figure 5 displays the comparison of accuracy on the MaleVis dataset. We can see from the figure that random forest performs somewhat better than the other methods. It may have trouble extrapolating outside of the training data set. A significant deviation of the test data from the training set of characteristics

could make the model's predictions less accurate. Although they may not be the most effective for extremely complex nonlinear patterns, they can capture nonlinear interactions.

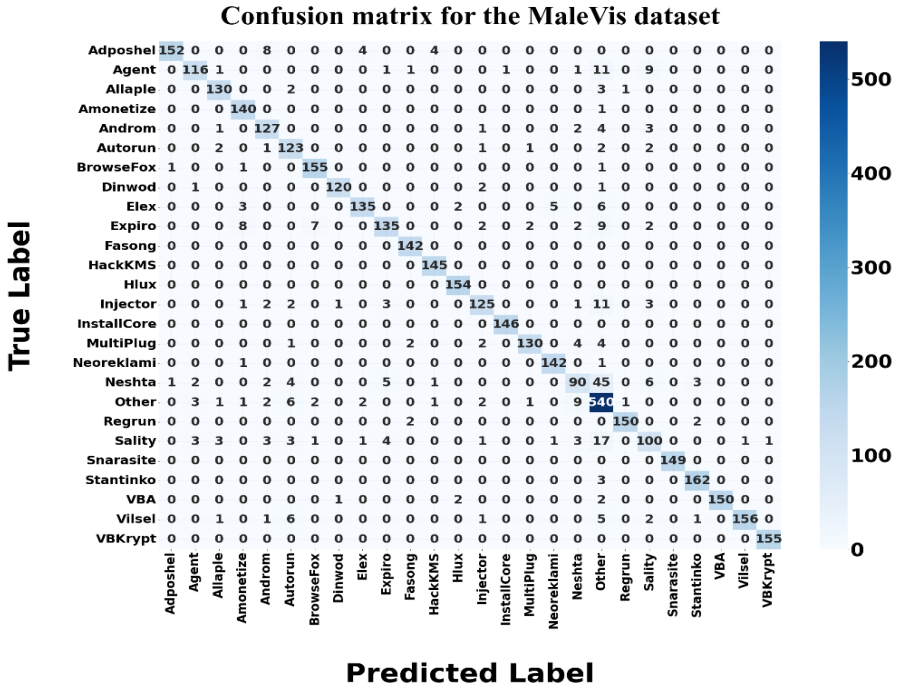

Figure 6. Confusion matrix on the MaleVis dataset

Second, the confusion matrix shown in Figure 6 was examined in conjunction with malware variations. It illustrated each malware variant's degree of accuracy. The rows in this matrix indicate the expected class, and the columns show the actual class. The number of accurately classified samples—that is, samples whose projected class matches their actual class—is displayed by the diagonal elements. Misclassified samples are represented by the off-diagonal elements. The proposed model accurately classified the majority of the data in each class, and the diagonal elements for all classes exhibit greater values than the off-diagonal elements. It demonstrates the suggested approach's effectiveness in classifying malware.

Performance evaluation on MMCC dataset

A comparison of the proposed model and existing method using the MMCC dataset is shown in this subsection. Finally, Table 4 displays the outcomes. The table indicates that our suggested approach's accuracy surpasses the current classification techniques, indicating the algorithm's considerable superiority over the most sophisticated approaches for malware family classification. It was accomplished with 99.26% accuracy, 99.22% precision, 99.24% sensitivity, 99.25% specificity, and 99.23% f1-score. The suggested framework's capacity to detect and classify malware is demonstrated by its higher values than other deep learning techniques already in use.

Table 2. Comparative analysis on the MMCC dataset

techniques	accuracy	precision	sensitivity	specificity	F1-score
MCFT-CNN (Kumar et al., 2021)	98.63	98.55	96	-	97.22
RNN (Gao et al., 2020)	96.90	96.92	96.90	-	96.81
Xception (Palma Salas et al., 2023)	98.22	98	96	-	97
MobileNet (Alnajim et al., 2023)	98.95	98.66	98.33	-	99
TCN (Sun et al., 2023)	96.15	-	-	-	95.96
Proposed	99.26	99.22	99.24	99.25	99.23

Additionally, figure 7 presents the accuracy and f1-score comparison on the MMCC dataset. Most deep learning techniques obtained accuracy levels above 98%, as seen in the image. Because deep learning recognizes high-level features from data level by level in concealed layers, it depicts the model's superior performance ratio above conventional machine learning methods. Certain characteristics are shared by malicious and benign samples, leading to incorrect categorization in conventional machine learning techniques. By producing high-level features, such characteristics can be diminished by using deep learning's hidden layers. Additionally, these techniques quickly pick up the high-level complicated features and can accurately distinguish between various attack types.

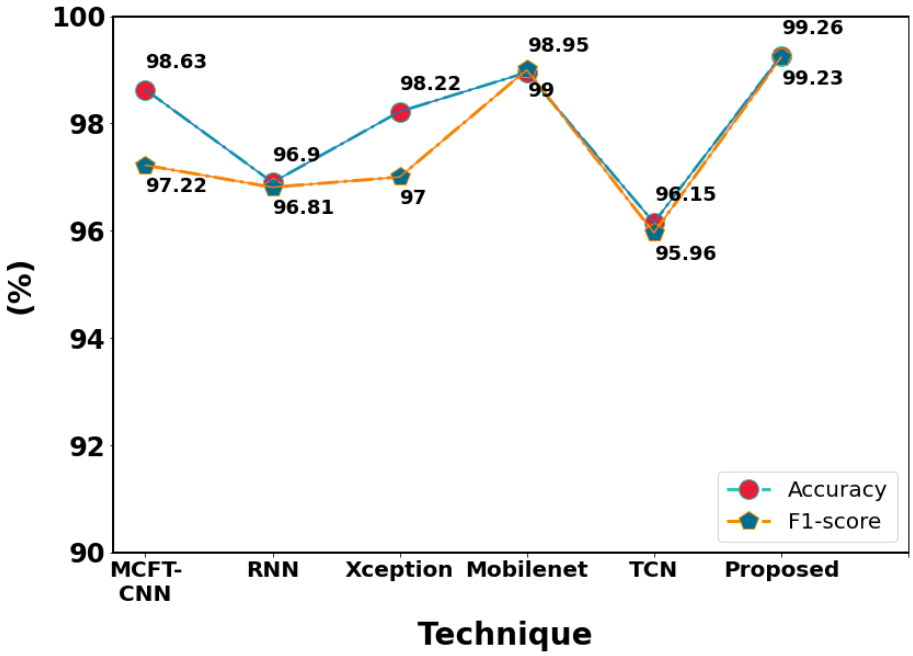

Figure 7. Accuracy and F1-score comparison on MMCC dataset

Moreover, the confusion matrix (Figure 8) displays the quantity of accurate and inaccurate predictions for every kind of malware assault. Classification values are represented

by the diagonal values, and misclassification is represented by the off-diagonal values. Given that the diagonal scores are more significant than the off-diagonal data and that there are relatively few misclassifications in the suggested model, it can be seen that it works well against most malware classes and accurately predicts the labels. It is made possible by deep Maxout networks and Res2Net's superior feature learning capabilities.

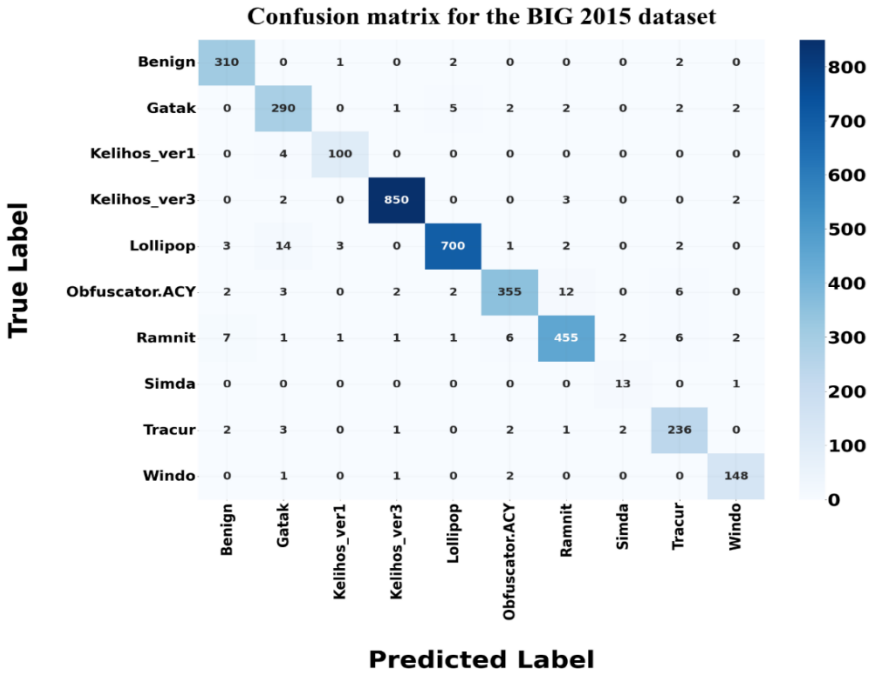

Figure 8. Confusion matrix on MMCC dataset

Performance evaluation on Drebin dataset

A comprehensive comparison was carried out on the Drebin dataset in order to evaluate the classification performance of the suggested technique. Table 3 displays the outcomes of each model's performance comparison. The results clearly show that the suggested model performs better than the current malware categorization techniques. The model demonstrates its efficacy in successfully categorizing malware by doing exceptionally well in multiple evaluation criteria, including 99.23% accuracy, 99.18% precision, 99.20% sensitivity, 99.22% specificity, and 99.19% f1-score. These striking outcomes demonstrate the potential and effectiveness of the suggested technique in the realm of malware identification. These results demonstrate the importance and influence of the study, presenting it as a viable approach to the problems related to successfully identifying and categorizing malware.

Table 3. Comparative analysis on the Drebin dataset

techniques	accuracy	precision	sensitivity	specificity	F1-score
Hypergraph neural networks (Zhang et al., 2023)	97.0	97.2	96.8	-	9699
Ensemble Decision Tree (EDT) (Al-Andoli et al., 2023)	95.4	94.3	97.3		95.7
CNN-LSTM (Hosseini et al., 2021)	96.5	96.4	97.9	-	97.7
KNN (Palma et al., 2024)	97.85				97.08
CNN (Atacak et al., 2022)	92	92.15	92	-	92.01
Proposed	99.23	99.18	99.20	99.22	99.19

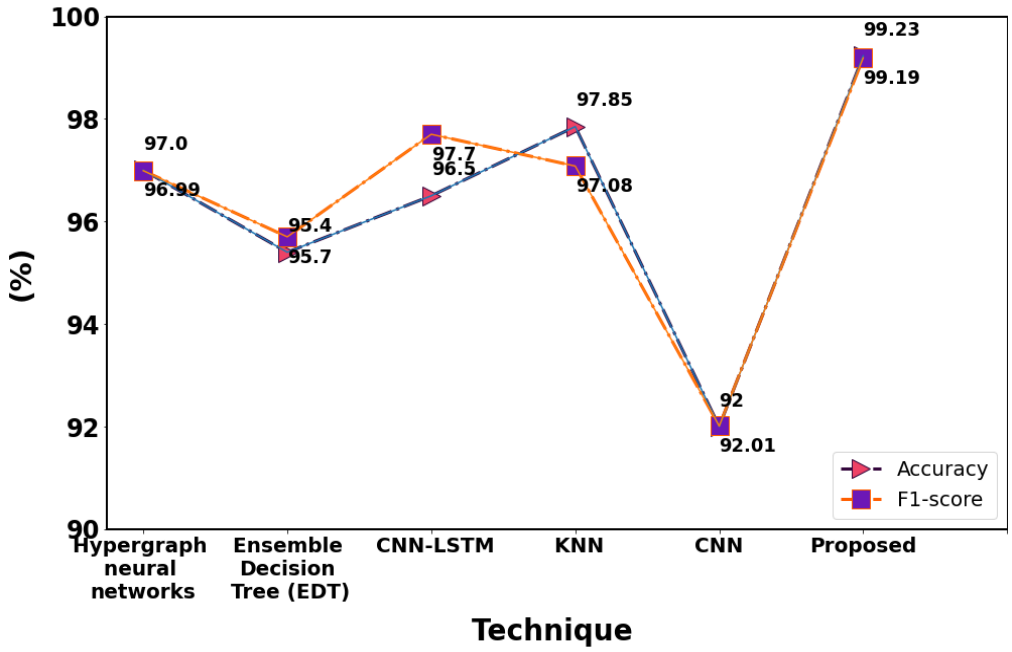

Figure 9. Accuracy and F1-score comparison on Drebin dataset

In contrast to alternative methods, CNN performs poorly. Figure 9 displays the 92% accuracy it obtained on the Drebin dataset. It happened as a result of the Drebin dataset's class imbalance. CNN may find it challenging to learn from minority classes if the dataset is unbalanced, with some classes being underrepresented. It would reduce the CNN's overall accuracy.

Additionally, the Drebin dataset contains more classes and fewer samples than the Microsoft dataset, complicating multi-classification. First, variants in malicious code families frequently show high levels of code repetition, which leaves slight variation between classes in the same family. Because of their innate resemblance, it is difficult for the model to accurately distinguish and categorize them. Nonetheless, the enhanced architecture of the proposed deep learning framework enables it to efficiently extract more intricate features from the input data, and the CGAN-based data augmentation technique resolves the imbalanced problem. This improved capacity makes it easier to distinguish and classify variants of the same harmful code family with greater accuracy.

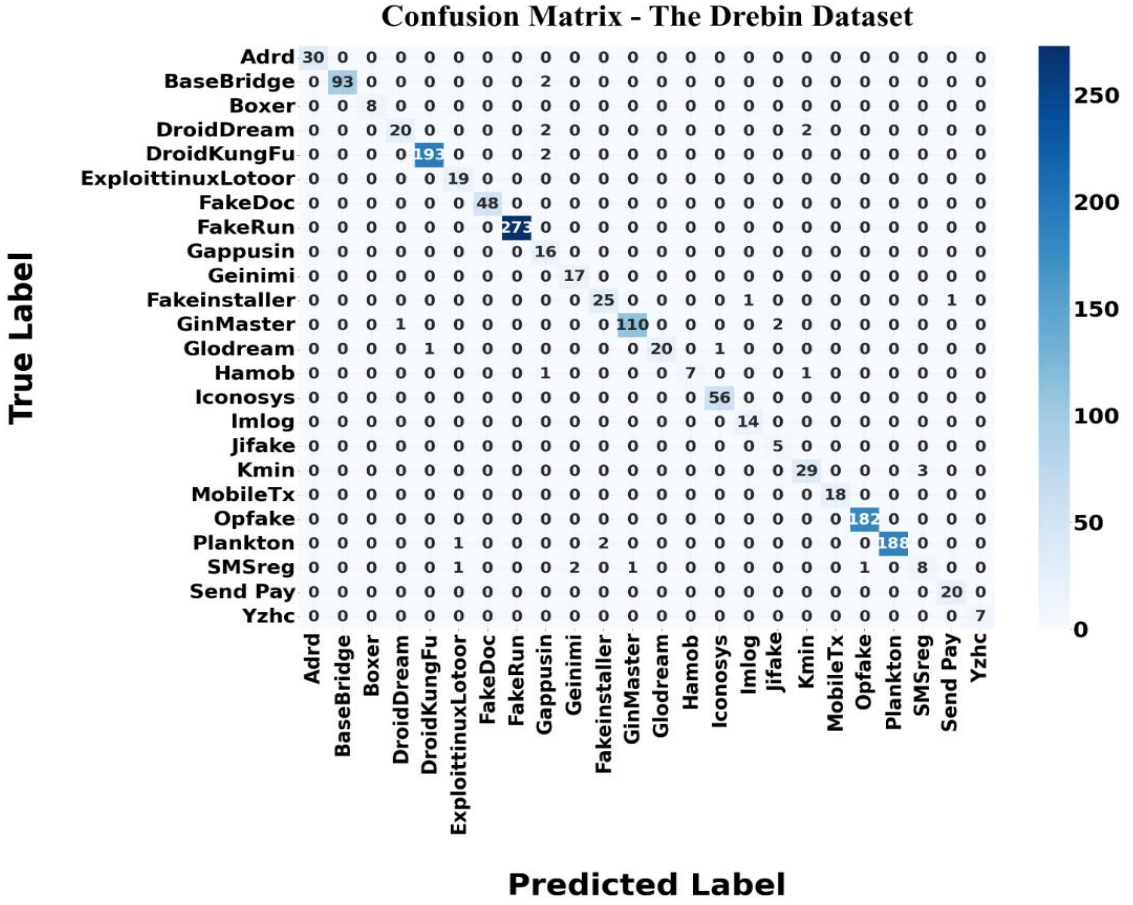

Figure 10. Confusion matrix on the Drebin dataset

Based on the test set's classification findings, the proposed framework's confusion matrix for Drebin is shown in Figure 10. Each row has the actual malware category, and each column contains the malware prediction category, allowing us to gauge how good our model is. The dataset is still incredibly unbalanced, as is evident. However, even with small families, the suggested framework achieves good accuracy. For example, it attains 100% accuracy on the Boxer and YZHC families, which have very few samples.

In summary, the aforementioned research indicates that the hybrid framework that has been suggested performs exceptionally well in classifying malware, outperforming other methods. Its increased performance will help to improve cybersecurity.

Conclusion

The cyber security domain continues to face severe danger from the ineffective recognition of malware versions despite extensive research on malware detection and classification. Malware identification is extremely difficult because of code obfuscation and packing methods. This research suggested a novel deep-learning architecture to identify malware variations efficiently. It integrates Deep Maxout network-based classification, Res2Net-based feature extraction, and CGAN-based data augmentation. In addition, COA is used to enhance the classifier's performance. The suggested framework outperformed the other approaches investigated in terms of classification accuracy, achieving better results for the MaleVis dataset (99.32%), the MMCC dataset (99.26%), and the Drebin dataset (99.23%) thanks to the combination of several valuable methodologies. Furthermore, the suggested model accurately detected most of the malware samples, demonstrating its resilience to the malware classification task. In the future, our goal is to create a more resilient system by fusing several deep learning architectures with reinforcement learning. Moreover, we can implement this framework with emerging technologies like big data, blockchain, and cloud computing to improve network security.

Acknowledgements

We declare that this manuscript is original, has not been published before and is not currently being considered for publication elsewhere.

Conflicts of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

The author’s received no financial support for the research, authorship, and/or publication of this article.

References

Ahmed, I. T., Jamil, N., Din, M. M., & Hammad, B. T. (2022). Binary and Multi-Class Malware Threads Classification. *Applied Sciences*, *12*(24), 12528.

Alamro, H., Mtouaa, W., Aljameel, S., Salama, A. S., Hamza, M. A., & Othman, A. Y. (2023). Automated android malware detection using optimal ensemble learning approach for cybersecurity. *IEEE Access*.

Al-Andoli, M. N., Sim, K. S., Tan, S. C., Goh, P. Y., & Lim, C. P. (2023). An ensemble-based parallel deep learning classifier with PSO-BP optimization for malware detection. *IEEE Access*.

Alnajim, A. M., Habib, S., Islam, M., Albelaihi, R., & Alabdulatif, A. (2023). Mitigating the risks of malware attacks with deep learning techniques. *Electronics*, *12*(14), 3166.

Alomari, E. S., Nuiaa, R. R., Alyasseri, Z. A. A., Mohammed, H. J., Sani, N. S., Esa, M. I., & Musawi, B. A. (2023). Malware detection using deep learning and correlation-based feature selection. *Symmetry*, *15*(1), 123.

Alzubi, O. A., Alzubi, J. A., Alzubi, T. M., & Singh, A. (2023). Quantum Mayfly optimization with encoder-decoder driven LSTM networks for malware detection and classification model. *Mobile Networks and Applications*, 1–13.

Aslan, Ö., & Yilmaz, A. A. (2021). A new malware classification framework based on deep learning algorithms. *IEEE Access*, *9*, 87936–87951.

Atacak, İ., Kılıç, K., & Doğru, İ. A. (2022). Android malware detection using hybrid ANFIS architecture with low computational cost convolutional layers. *PeerJ Computer Science*, *8*, e1092.

Atitallah, S. B., Driss, M., & Almomani, I. (2022). A novel detection and multi-classification approach for IoT-malware using random forest voting of fine-tuning convolutional neural networks. *Sensors*, *22*(11), 4302.

Aurangzeb, S., & Aleem, M. (2023). Evaluation and classification of obfuscated Android malware through deep learning using ensemble voting mechanism. *Scientific Reports*, *13*(1), 3093.

Chaganti, R., Ravi, V., & Pham, T. D. (2022). Image-based malware representation approach with EfficientNet convolutional neural networks for effective malware classification. *Journal of Information Security and Applications*, *69*, 103306.

Chen, Z., & Cao, J. (2023). VMCTE: Visualization-based malware classification using transfer and ensemble learning. *Computers, Materials & Continua*, *75*(2).

Das, S., Garg, A., & Kumar, S. (2024). Stacking ensemble-based approach for malware detection. *SN Computer Science*, *5*(1), 185.

Demirkıran, F., Çayır, A., Ünal, U., & Dağ, H. (2022). An ensemble of pre-trained transformer models for imbalanced multiclass malware classification. *Computers & Security*, *121*, 102846.

Dhalaria, M., & Gandotra, E. (2024). MalDetect: A classifier fusion approach for detection of Android malware. *Expert Systems with Applications*, *235*, 121155.

El-Ghamry, A., Gaber, T., Mohammed, K. K., & Hassanien, A. E. (2023). Optimized and efficient image-based IoT malware detection method. *Electronics*, *12*(3), 708.

Eslavath, R., & Mummadi, U. K. (2024). ENSIC: Feature selection on Android malware detection attributes using an enhanced nonlinear SVM integrated with cross validator. *International Journal of Intelligent Systems and Applications in Engineering*, *12*(2), 495–504.

Falana, O. J., Sodiya, A. S., Onashoga, S. A., & Badmus, B. S. (2022). Mal-Detect: An intelligent visualization approach for malware detection. *Journal of King Saud University – Computer and Information Sciences, 34*(5), 1968–1983.

Gao, X., Hu, C., Shan, C., Liu, B., Niu, Z., & Xie, H. (2020). Malware classification for the cloud via semi-supervised transfer learning. *Journal of Information Security and Applications, 55*, 102661.

Hemalatha, J., Roseline, S. A., Geetha, S., Kadry, S., & Damaševičius, R. (2021). An efficient DenseNet-based deep learning model for malware detection. *Entropy, 23*(3), 344.

Hosseini, S., Nezhad, A. E., & Seilani, H. (2021). Android malware classification using convolutional neural network and LSTM. *Journal of Computer Virology and Hacking Techniques, 17*(4), 307–318.

Jiang, J., & Zhang, Y. (2023). A pyramid stripe pooling-based convolutional neural network for malware detection and classification. *Journal of Ambient Intelligence and Humanized Computing, 14*(3), 2785–2796.

Kim, J., Paik, J. Y., & Cho, E. S. (2023). Attention-based cross-modal CNN using non-disassembled files for malware classification. *IEEE Access, 11*, 22889–22903.

Kumar, S. (2021). MCFT-CNN: Malware classification with fine-tune convolution neural networks using traditional and transfer learning in Internet of Things. *Future Generation Computer Systems, 125*, 334–351.

Kumar, S., Janet, B., & Neelakantan, S. (2024). IMCNN: Intelligent malware classification using deep convolution neural networks as transfer learning and ensemble learning in honeypot-enabled organizational network. *Computer Communications.*

Liu, Y., Yang, P., Jia, P., He, Z., & Luo, H. (2022). MalFuzz: Coverage-guided fuzzing on deep learning-based malware classification model. *PLOS ONE, 17*(9), e0273804.

Maray, M., Alqahtani, H., Alotaibi, S. S., Alrayes, F. S., Alshuqayran, N., Alnfiai, M. M., & Al Duhayyim, M. (2023). Optimal bottleneck-driven deep belief network enabled malware classification on IoT-cloud environment. *Computers, Materials & Continua, 74*(2).

Nikam, U., & Deshmukh, V. M. (2024). Hybrid feature selection technique to classify malicious applications using machine learning approach. *Journal of Integrated Science and Technology, 12*(1), 702–702.

Niknami, Omid & Akhondzadeh Noughabi, Elham (2024). Dynamic Pricing of Customer Classes in Rail Transportation Systems Using Deep Q Network Algorithm. *Industrial Management Journal*, 16(4), 597-630.

Othman, H., AlHija, M. A., & Alsharaiah, M. A. (2024). Toward enhancing malware detection using practical swarm optimization in honeypot. *International Journal of Intelligent Engineering & Systems, 17*(1).*

Palma Salas, M. I., De Geus, P., & Botacin, M. (2023, October). Enhancing malware family classification in the Microsoft challenge dataset via transfer learning. In *Proceedings of the 12th Latin-American Symposium on Dependable and Secure Computing* (pp. 156–163).

Palma, C., Ferreira, A., & Figueiredo, M. (2024). Explainable machine learning for malware detection on Android applications. *Information, 15*(1), 25.

Parihar, A. S., Kumar, S., & Khosla, S. (2022). S-DCNN: Stacked deep convolutional neural networks for malware classification. *Multimedia Tools and Applications, 81*(21), 30997–31015.

Ravi, V., & Alazab, M. (2023). Attention-based convolutional neural network deep learning approach for robust malware classification. *Computational Intelligence, 39*(1), 145–168.

Sharma, O., Sharma, A., & Kalia, A. (2024). MIGAN: GAN for facilitating malware image synthesis with improved malware classification on novel dataset. *Expert Systems with Applications, 241*, 122678.

Son, T. T., Lee, C., Le-Minh, H., Aslam, N., & Dat, V. C. (2022). An enhancement for image-based malware classification using machine learning with low dimension normalized input images. *Journal of Information Security and Applications, 69*, 103308.

Sun, J., Luo, X., Wang, W., Gao, Y., & Zhao, W. (2023). Robust malware identification via deep temporal convolutional network with symmetric cross entropy learning. *IET Software, 17*(4), 392–404.

Tang, Y., Qi, X., Jing, J., Liu, C., & Dong, W. (2023). BHMDC: A byte and hex n-gram-based malware detection and classification method. *Computers & Security, 128*, 103118.

Tekerek, A., & Yapici, M. M. (2022). A novel malware classification and augmentation model based on convolutional neural network. *Computers & Security, 112*, 102515.

Yadav, B., & Tokekar, S. (2023). Malware multi-class classification based on malware visualization using a convolutional neural network model. *International Journal of Information Engineering and Electronic Business, 13*(2), 20.

Zhan, D., Hu, Y., Li, W., Chen, J., Guo, S., & Pan, Z. (2023). Towards robust CNN-based malware classifiers using adversarial examples generated based on two saliency similarities. *Neural Computing and Applications*, 1–18.

Zhang, D., Wu, X., He, E., Guo, X., Yang, X., Li, R., & Li, H. (2023). Android malware detection based on hypergraph neural networks. *Applied Sciences, 13*(23), 12629.

Zou, B., Cao, C., Wang, L., Fu, S., Qiao, T., & Sun, J. (2024). FACILE: A capsule network with fewer capsules and richer hierarchical information for malware image classification. *Computers & Security, 137*, 103606.

Bibliographic information of this paper for citing:

Deol, G. Joel Sunny; Prasad, Kanakam Siva Rama; Hanumat Prasad, Alahari; Rao, M. Chennakesava; Alekhya, Medida & Alekhya, T. (2026). DeepSeek vs. ChatGPT: RES2NET-MAXOUT: Image-Based Multi-Class Malware Classification with Deep Learning Models. *Journal of Information Technology Management*, 18 (3), 1-24.
https://doi.org/10.22059/jitm.2026.388431.3967