ISSN: 2182-2069 / E-ISSN: 2182-2077

An Efficient Malware Identification Model Using a GWO-Tuned CNN with Asynchronous Distributed Learning

Sadeq Thamer Hlama[1], Abolfazl Gandomi[2*], Zuhair Hussein Ali[3], and Mohammadreza Soltanaghaei[4]

[1]PhD Candidate, Department of Computer Engineering, Institute of Artificial Intelligence and Social and Advanced Technologies, Isf.C., Islamic Azad University, Isfahan, Iran.
sadeqthamer1976@gmail.com, https://orcid.org/0000-0002-9697-4736

[2*]Department of Computer Engineering, Ya.C., Islamic Azad University, YAZD, Iran.
abolfazl.gandomi@iau.ac.ir, https://orcid.org/0000-0003-3353-3762

[3]Department of Computer Science, College of Education, Mustansiriyah University, Iraq.
zuhair72h@uomustansiriyah.edu.iq, https://orcid.org/0000-0002-9380-5961

[4]Department of Computer Engineering, Institute of Artificial Intelligence and Social and Advanced Technologies, Isf.C., Islamic Azad University, Isfahan, Iran. soltan@iau.ac.ir,
https://orcid.org/0000-0002-2930-9066

Received: February 28, 2025; Revised: April 08, 2025; Accepted: May 13, 2025; Published: May 30, 2025

Abstract

In light of the changing landscape of cybersecurity, accurate malware detection and identification remain relevant issues. The following paper presents a new malware detection model for optimizing detection efficiency and scalability with the Grey Wolf Optimizer (GWO) algorithm and Distributed Convolutional Neural Networks (CNNs). Apart from improving the scalability of the system, the suggested distributed architecture increases its adaptability to new and advanced malware risks. Experimental evaluation of the proposed framework using the Malimg benchmark dataset demonstrates substantial enhancements in key performance indicators such as accuracy, precision, recall, and F-score. Achieving a remarkable 99.95% accuracy, the framework underscores its reliability and practical applicability in real-world cybersecurity contexts. These results confirm the efficacy of metaheuristic optimization methods in combination with deep learning algorithms in solving the complexity of modern malware detection. This contribution enhances the state of the art in cybersecurity by providing a systematic and scalable solution to the ever-changing nature of malware threats.

Keywords: Malware Detection, Grey Wolf Optimizer, Distributed Convolutional Neural Networks, Metaheuristic Optimization, Cybersecurity, Malimg Dataset, Deep Learning.

1 Introduction

The spread of interconnected systems in the digital age has revolutionized how individuals, organizations, and nations conduct their operations and allow previously unheard-of levels of efficiency and connection. Despite technological progress, it has also created favorable conditions for increasingly sophisticated cybersecurity threats, with malware ranking among the most prevalent and harmful forms

Journal of Internet Services and Information Security (JISIS), volume: 15, number: 2 (May), pp. 827-844.
DOI: *10.58346/JISIS.2025.I2.055*

*Corresponding author: Department of Computer Engineering, Ya.C., Islamic Azad University, YAZD, Iran.

(Hossain & Islam, 2024). Malware encompasses a broad spectrum of malicious software, such as viruses, worms, trojans, ransomware, and spyware, with each having a different attack method and tactics of subjugating systems and information (Gaber et al., 2024). This has been coupled with the fast development of malware using techniques like polymorphism and code obfuscation making it even harder to detect and mitigate thus raising vital concerns with respect to the stability and security of the digital infrastructures globally (Smmarwar et al., 2024; Nawshin et al., 2024).

Malware detection has largely been concerned with static analysis and signature methods where a threat is identified by examining the code patterns or known patterns in known malware instances (Alavala et al., 2025). Although these approaches are proficient at recognizing known threats, they frequently struggle to identify new or concealed forms of malware, thereby exposing systems to zero-day vulnerabilities. To address this limitation, dynamic or behavioral analysis has emerged as an alternative approach, monitoring program execution to identify malicious activity based on operational patterns (Dahiya et al., 2025). Nevertheless, this method is not flawless, either, because it is likely to result in high false positives rates because of the overlapping nature of benign and malicious behaviors in complex settings (Redhu et al., 2024). The drawbacks of the conventional detection approaches have led to a new phase where state-of-the-art machine learning (ML) and deep learning (DL) methods are used (Hasan et al., 2025; Armstrong & Tanaka, 2025). Particularly, deep learning has shown to have a great potential in malware identification (Nguyen et al., 2025). Convolutional Neural Networks (CNNs) have shown to be of considerable potential as a deep learning method for visualizing binary samples of malware, for example, grayscale images, and hierarchically extracting visualized samples automatically (Kim et al., 2022; Ko et al., 2025). As opposed to earlier machine learning techniques that demand substantial manual intervention for designing input feature, CNNs can autonomously learn subtle, previously unseen malicious patterns, making them a powerful tool for malware detection (Bakır, 2025; Wasif et al., 2025). Despite these advantages, deploying DL models for malware detection is not without challenges. Issues such as hyperparameter optimization and computational scalability present significant hurdles (Perera & Wickramasinghe, 2024; Revathy et al., 2025). The inefficient hyperparameters would result in overfitting, long training, or low detection rates, and the computational resources-hungry DL models make them difficult to apply to practical settings, especially when they are handling large-scale malware data (Alzaben et al., 2025; Mouhtadi et al., 2024).

This paper introduces a new architecture for combining distributed deep learning models with metaheuristic optimization techniques to solve such problems (Luedke & Kingdone, 2025). With its balance between exploitation and exploration of search spaces within high-dimensional search spaces, i.e., the Grey Wolf Optimizer (GWO) algorithm, optimizes CNN hyperparameters (Ferdous et al., 2025). According to grey wolves' hierarchical prey hunting behavior, the GWO algorithm tunes parameters like filter size, learning rate and network depth, thus boosting the general performance of convolutional neural networks. Additionally, the computational bottlenecks are overcome through adopting a distributed strategy for CNNs, i.e., Fault tolerance is achieved by dividing the task of malware detection task among multiple concurrent local CNNs. Scalability and handling massive-data are also successfully attained through distribution. Utilization of Asynchronous Distributed Stochastic Gradient Descent (ADSDG) even further increases the flexibility of the system because it enables it to learn dynamically from multiple and ever-altering sources of data yet remain robust in responding. Combined, they will assist in having a scalable, accurate, and effective answer to the increasing complexity of malware detection in advanced digital contexts.

The major contributions of this study are as follows:

- Utilizes GWO to optimize CNN hyperparameters, enhancing malware classification performance. Balances exploitation and exploration by continuously adjusting influence of each wolf on others' decisions.
- Investigates new search space areas and prevents local optimum by introducing random perturbations to wolves' positions. Increases computational capability and system performance by allowing parallel and distributed processing.
- Deploys a group of local CNNs as worker nodes to learn parameters from a master server. Provides fault tolerance and scalability through scaling learning rates in a group of independent local networks.
- Periodic, frequent weight updates during one-to-many local training enhance model global accuracy and efficiency in learning. Reduces oscillations caused by local updates to obtain converged models at one of the global optima.

The rest of the paper is structured as follows: In section 2 we review related works and point out the gaps and challenges that still exist in the area of malware identification. The detailed description of the suggested approach is given in section 3. Section 4 provides the outcomes of the conducted experiment and comments on the effectiveness of the offered framework. Finally, Section 5 will close with a conclusion of the most significant conclusions of the study, along with potential themes of future research.

2 Related Works

The paper (Qureshi et al., 2024) gives an in-depth survey of deep learning-based IoT malware detection and forensics (Krishnan & Singh, 2021). Based on the type of available research, the authors divide it into four major topics: deep learning, malware forensics, anti-forensics and IoT security (Rathi et al., 2024). In one sense, the power of deep learning in enhancing the security of IoT is unveiled; in another, the survey showed a laundry list of research gaps required like there being no humongous IoT-specific data-sets, having to do interdisciplinary research, or devising sophisticated countermeasures against anti-forensic attacks Reddy et al., (2025). In (Jiang et al., 2024), an emerging hybrid malware classification and analysis method is discussed where the Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs) are utilized. This approach maps binary images to grayscale images so that image processing operations can be applied for analysis. These processed images are then analyzed using a CNN-LSTM model to analyze both spatial and temporal features. PCA is used to select the most important features. With the aim to improve the accuracy of classification, voting is employed to aggregate predictions of more than one classifier. The method has a detection rate of other malware families lower than other existing methods.

In (Liu et al., 2024), VBDN, a new malware detector framework, is presented. The framework provides CNN-based usage for best malware classification and employs various methods for optimization of performance such as data visualization, utilization of balanced data, data augmentation, and deep CNN structure. VBDN provides over 90% accuracy in the classification task by using four open-source datasets. A confusion matrix provides some statistics about how each class of malware has been classified. The balanced data adoption strategy avoids creating new datasets, which causes a huge reduction in the computational cost. Furthermore, data augmentation tricks are used in order to get the model generalized. VBDN has the optimal trade-off accuracy and computational complexity compared to some of the latest techniques and offers an efficient implementation of malware detection (Nandy et al., 2025). An innovative approach is presented for improving malware detection and classification

through the efficient processing of dynamic and static features in (Zhang et al., 2025). The static properties are encoded as images and the dynamic properties are fixed-length sequence transformed. These preprocessed properties are used to train an AI model that can effectively detect and classify malware. The experiments confirm the 99.34% accuracy of malware detection and 95.1% accuracy of malware classification with the fast prediction time of approximately 0.1 seconds per sample. The results indicate the capability of the system to increase malware detection and classification significantly. The CNN-LSTM-based malware detection system is presented in the article (Qi et al., 2023) for detecting malicious behavior. In order to address the weaknesses of the previous statical analysis methodologies, this system is concentrated on dynamic analysis, which gains the important indicators of behavior by parsing logs, monitoring API calls, and extension checking. The system attains a remarkable accuracy rate of 96%, which means that the system can be used to dependably detect malicious activities.

A recent study proposed a malware classification framework utilizing deep learning techniques, particularly convolutional neural networks (CNNs) and transfer learning, achieving a classification accuracy of 99% with models like ResNet-50 and MobileNetV2. This framework addresses challenges such as overfitting and computational costs while enhancing robustness against adversarial attacks, although vulnerabilities to sophisticated attack vectors remain a concern (source (Bakır, 2025)). Additionally, Al-Khater et al. (Al-Khater & Al-Madeed, 2024) introduced a hybrid model combining VGG and ResNet architectures with a Feature Adaptive Bidimensional Empirical Mode Decomposition (FABEMD) module, which effectively extracts features and improves classification accuracy by recognizing subtle data pattern differences Bhuiyan et al., 2008, Shaukat et al., 2024, introduced a malware detection framework based on RegNetY320, utilizing a One-Class Support Vector Machine (OCSVM) for classification and incorporating a feature selection step to enhance efficiency by reducing redundancy. Singh et al., 2024, developed a hybrid model that combines Gated Recurrent Units (GRU) and Convolutional Neural Networks (CNN) with Class-Specific Bandwidth (CSBW) features and a Random Forest (RF) classifier, improving malware detection through enhanced temporal and spatial feature learning. Behera et al., 2025, proposed a two-stream deep learning framework that leverages ResNet50 and VGG16 networks to improve classification accuracy by fusing distinct feature extraction levels, effectively addressing the variability in malware features. A similar but effective framework was used by Kumar et al., 2025, which consisted of the ResNet50 model and a standard CNN. In this way, the essence of the method is to use the possibilities of deep residual learning offered by ResNet50, but to supplement it with conventional CNN layers to extract more features. Being simpler than other hybrid models, it still shows competitive results and can be taken as a good basis of improvements.

3 Proposed Method

The study is on the design of an effective malware detection system on distributed deep CNNs with Grey Wolf Optimizer tuned CNN hyperparameters. The two steps of the provided process are CNN hyperparameter tuning and classification using a distributed CNN. Determining the best network architecture in CNNs is challenging because CNNs possess complicated architecture, coupled parameters, and extended training times. The GWO algorithm also optimizes the initial best set of hyperparameters for the CNN in the first stage. GWO is a grey wolf metaheuristic that is based on hierarchical hunt strategy of grey wolves. Its exploitation and exploration ability facilitates efficient global optimum discovery and hence is most critical in hyperparameter tuning where efficient malware detection relies on optimal configuration to be triggered. The creation and implementation of a distributed framework for malware classification follows as the subsequent step. The distributed CNN network follows from the best achieved hyperparameters in the first step. In the proposed distributed

setting, there will be greater than one local convolutional network that classify malware in parallel with common parameters and synchronized and updated from time to time via a master node central server. All of the above is learned using a distributed asynchronous stochastic gradient descent (ADSDG) algorithm. ADSGD informs all the worker nodes participating in learning of the learned system so that the model is able to incrementally build upon random subsets of the training set. Besides, the system is made scalable through independent execution of the local networks and synchronized weights update and learning rate, thereby enhancing its performance in malware emerging threats. Figures 1 illustrates the structure of the proposed approach.

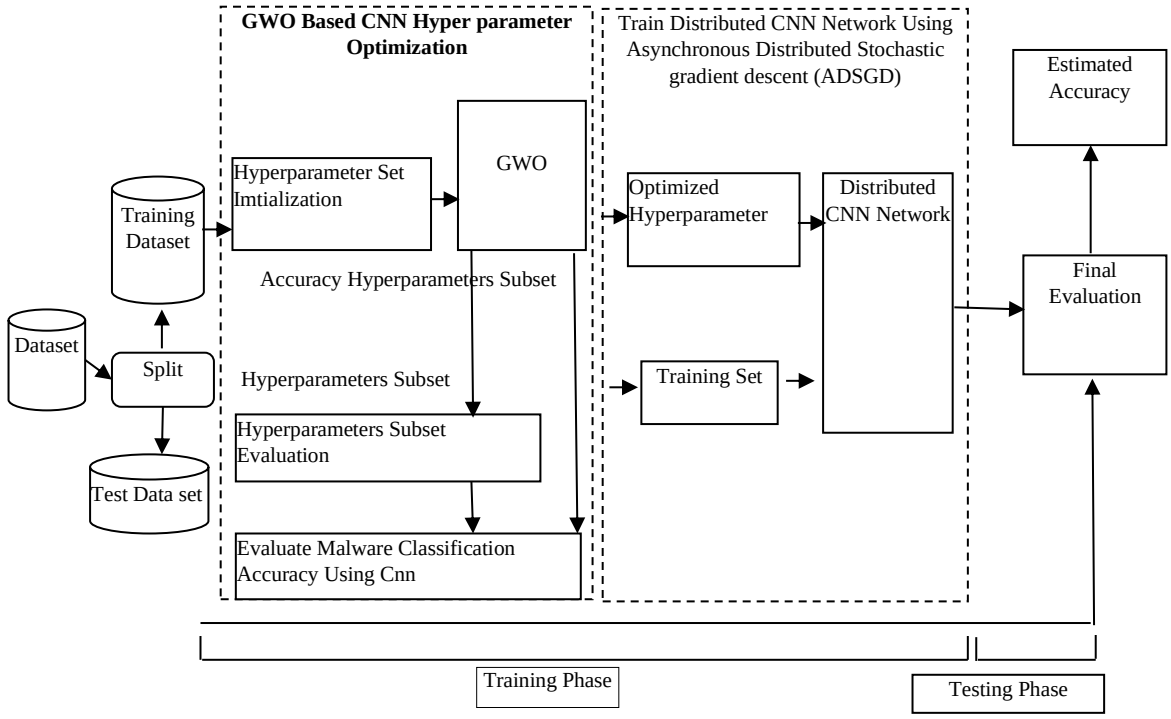

Figure 1: The Suggested Model Diagram

3.1. Utilizing the GWO to Optimize CNN Hyperparameter

The CNNs are very robust deep learning-based methods that have been very promising in different applied areas. Although it is difficult and exhausting to achieve an optimal CNN configuration due to the complicated network topology, dependence on hyperparameters, and long training time. Grey Wolf Optimizer (GWO) algorithm is employed here in attempting to efficiently solve the issue of hyperparameter optimization. The GWO algorithm is employed initially in finding optimal CNN hyperparameters, which are later applied in the network to train the network. The fitness of the GWO algorithm is the network's accuracy. The GWO algorithm also optimizes parameters based on the fitness values in the first step. This recursive process is repeated until a halting condition exists. Hyperparameter tuning process for CNNs of the GWO algorithm is shown in Figure 2.

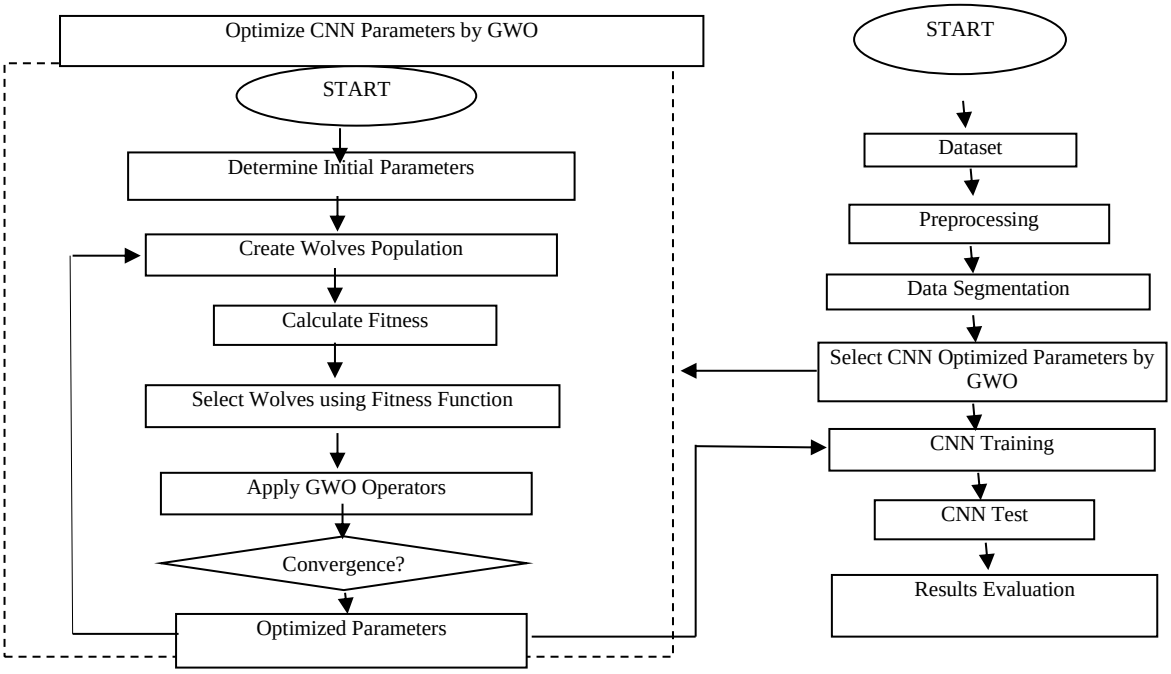

Figure 2: CNN Hyperparameter Optimization Using GWO

As can be seen earlier, in this research the hyperparameters of CNN are optimized employing the Grey Wolf Optimizer approach in best possible manner. Each 'wolf' in book algorithm corresponds to a set of CNN hyperparameters, i.e., learning rate, number of layers, number of neurons, filter size. Table 1 has complete list of optimized CNN hyperparameters in this work.

Table 1. List of CNN Hyperparameters to Optimize

Hyperparameter No.	Hyperparameter Description	Allowed hyperparameters values
1	Kernel numbers in 1st Conv layer	[1, 300]
2	Size of kernel in 1st Conv layer	{9×9, 7×7, 5×5, 3×3}
3	Activation function in 1st Conv layer	{Tanh, Sigmoid, ReLU}
4	Kernel numbers in 2nd Conv layer	[1, 300]
5	Learning rate scale in 1st Conv layer	{0, 10}
6	Size of kernel in 2nd Conv layer	{9×9, 7×7, 5×5, 3×3}
7	Activation function in 2nd Conv layer	{Tanh, Sigmoid, ReLU}
8	Scale of learning rate in 2nd Conv layer	{0, 10}
9	Neurons number in 1st FC layer	[10, 800]
10	Activation function in 1st FC layer	{Tanh, Sigmoid, ReLU}
11	Dropout ratio in 1st FC layer	[0.1, 0.9]
12	Neurons number in 2nd FC layer	[10, 800]
13	Activation function in 2nd FC layer	{SoftMax, Sigmoid, ReLU}
14	Dropout ratio in 2nd FC layer	[0.1, 0.9]
15	Learning rate	[0.001, 1]

Grey Wolf Optimizer (GWO) technique used in this research has procedures step by step as follows:

Step 1: Initialization

This Step creates the initial population of wolves. Every wolf stands in for a specific of network hyperparameters. For every parameter, these hyperparameters are started randomly as A_i where, A_i

shows the I-th agent's position in the solution domain GWO Algorithm. we can denote $A_i = [a_1.a_2.\ldots.a_d]$. here d stands for the hyperparameter count. The network components and their corresponding allowed bounds are detailed in Table 1.

Step 2: Fitness Evaluation

Each wolf's fitness score is calculated based on how well the CNN performs when using the hyperparameters represented by that particular wolf. In this approach, accuracy serves as fitness criterion, computed as follows:

$$Fitness\_Fun = \frac{(TP + TN)}{(TP + FP + TN + FN)} \quad (1)$$

Step 3: Rank the wolves by their positions

Based on their ranking performance in the last step, alpha (α), beta (β), and delta (δ) are three ranks into which the wolves divide. They are ranked as leaders of a pack where the highest-ranking wolf becomes α, the second-ranked one becomes β, and the third one is referred to as δ. The rest of the wolves are labeled the omega (ω) tag and get their locations adjusted based on α, β, and δ.

Step 4: Position Updating

each wolf updates its position by gravitating toward the leading wolves—α, β, and δ. The proximity to each leader is computed using the following equations:

$$D_\alpha = |C_1 A_\alpha - A| \quad (2)$$

$$D_\beta = \left|C_1 A_\beta - A\right| \quad (3)$$

$$D_\delta = |C_1 A_\delta - A| \quad (4)$$

Here, random vectors computed as C_1, C_2, and C_3 are found in:

$$C = 2r \quad (5)$$

In this case, r random number between 0 and 1.

The updated position of a wolf is determined by averaging the influence exerted by the δ, β, and α leaders.

$$A_{new} = \frac{A_1 + A_2 + A_3}{3} \quad (6)$$

Here, A_1, A_2, and A_3 are computed as:

$$A_1 = A_\alpha - P_1 D_\alpha \quad (7)$$

$$A_2 = A_\beta - P_2 D_\beta \quad (8)$$

$$A_3 = A_\delta - P_3 D_\delta \quad (9)$$

Where, the control parameter vectors P_1, P_2, and P_3 are calculated as follows:

$$P = 2\gamma r - \gamma \quad (10)$$

Factors γ and r, derived from the method's constants, govern the convergence behavior. Over iterations, γ lowers gradually from two to zero. Also, r is a random vector falling between 0 and 1.

Step 5: Termination Criterion

Steps 2 to 4 are repeated until the stopping condition is met—which in our case is iterations.

3.2. Distributed CNN Architecture for Malware Identification

This paper classifies and detects different types of malware with a distributed CNN structure. Different local convolutional networks—separate workers—are responsible for malware classification and learning weights through central repository server in the proposed system. The failure of a LAN (Local Area Network) does not impact the effectiveness of the integrated system. In addition, network scalability is supported by the modification of the learning rate through independently operating local networks, thus enabling improved detection of new malware. Thus, detection accuracy by the network is improved. Usually, the distributed CNN works as follows: in initial step, the malware dataset is split across N subsets, where N stands for the slave node count. Every element of the data finds a slave node. In every iteration, all the slave nodes calculate the error gradient and use the master node to label the given data. The master node gets these error gradients afterward and adjusts the network weights based on their obtained value. The slave nodes get the updated weights back-through. It goes on until the number of given iterations are done. Figures 3 show the distributed convolutional network architecture.

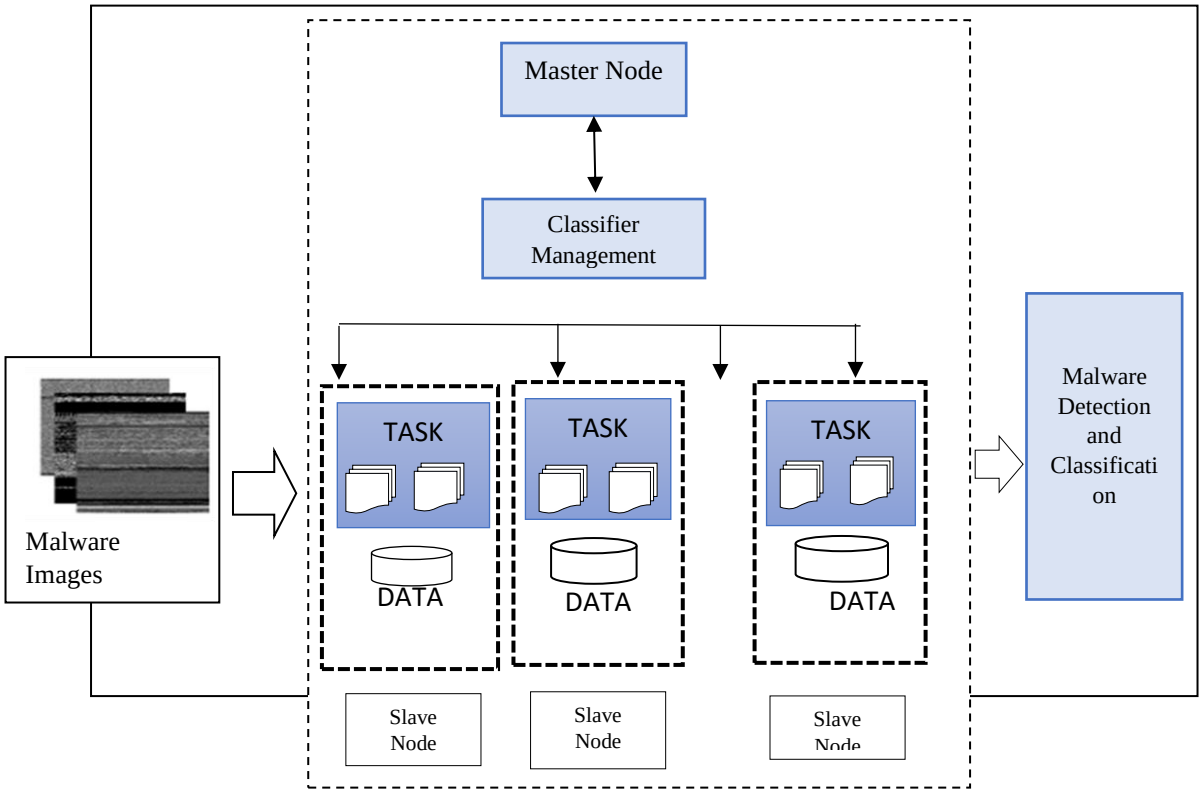

Figure 3: Structure of Distributed Convolutional Network

3.2.1. Learning Strategy and Parameter Updating Mechanism for Distributed Convolutional Network

This study trains the distributed convolutional neural network (CNN) with the Asynchronous Distributed Stochastic Gradient Descent (ADSDGD) algorithm. Maximum Likelihood directed classification is used

in training set classification in the proposed method. ADSGD is more efficient in bringing learned models to the entire involved worker nodes in the event of completely distributed sets of data. In terms of the factor that it generates random data from the training set, the model incrementally moves. Unlike Synchronous Stochastic Gradient Descent (SSGD), in which every device has the ability to fault stop the whole learning process, ADSGD is fault-tolerant. Whenever a worker goes offline, the other workers proceed with the learning process and synchronize parameter updates using the assistance of a central server. Unlike SSGD, where all nodes are synchronized and cause global delays when a node goes down, ADSGD is fault-tolerant because the worker nodes are independent. Figure 4 illustrates distributed CNN's parameter update mechanism. Figure 4 illustrates how each of the worker nodes talks to the master node's global parameter server. The updates from workers, i.e., learning rate scaled gradients, get pushed onto the master repository server. The parameter updates in the proposed model are according to the following relation:

$$W_{i+1} = W_i - \lambda \sum_{j=1}^{N} \Delta W_{i.j} \quad (11)$$

Here, ΔW_(i, j) is the primary update vector; λ is the factor of scaling. In standard SGD training, parameter vector W will be calculated as iteratively i+1:

$$W_{i+1j} = W_i - a\nabla L_j \quad (12)$$

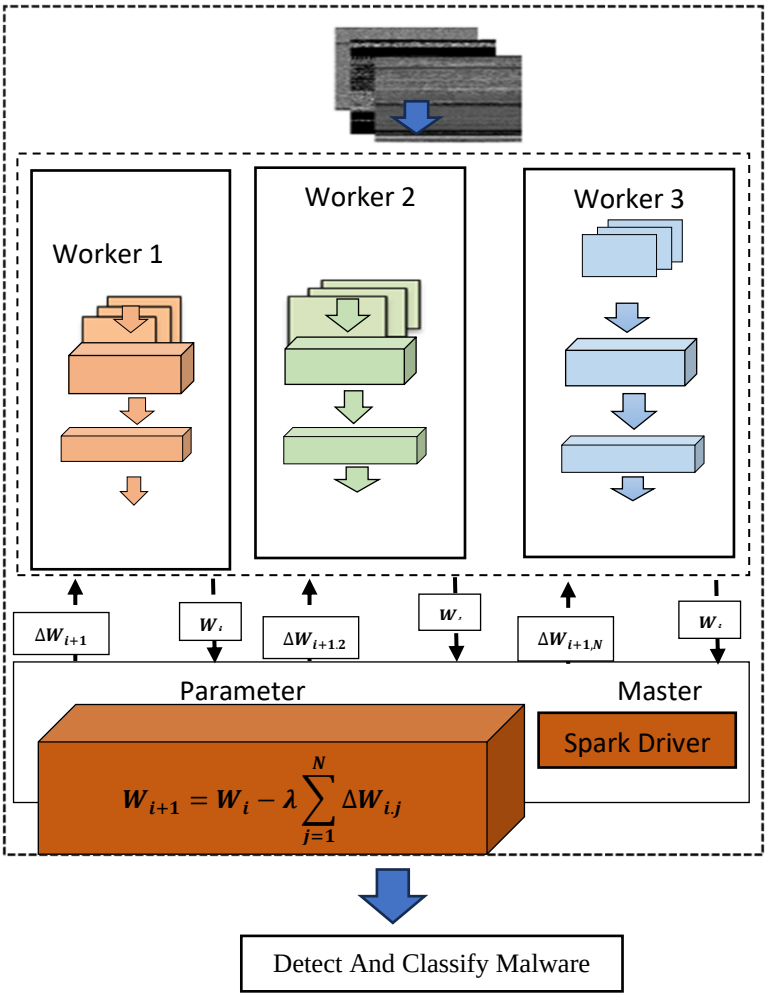

Figure 4: Distributed CNN Parameter Update Procedure

Wherein, L_j represents the loss function; a denotes the learning rate; hence, ∇L_j is obtained as:

$$\nabla L_j = \left(\frac{\partial L}{\partial w_i} \ldots . \frac{\partial L}{\partial w_n}\right). for\ n\ parameters \quad (13)$$

ADSGD is defined by the direct utilization of the parameter updates ΔW_(i.j) directly in the parameter vector W, in a manner that allows the workers to utilize the latest parameter updates once computed, in order to accelerate and improve training. Worker node parameter update based on ADSGD is shown in Algorithm 1 and Algorithm 2 represents the handling of updates on the master node.

Algorithm 1: ADSGD pseudocode in Workers side

Inputs: learning rate alpha, Convolutional structure, Workers E $\{e_1 \ldots\ldots e_n\}$ Outputs: partial gradient ΔW_i
1. To every worker in set E use 2. Reading master node (server parameter) information 3. Selecting a random sample from s ∈ {1, ..., b} in local data 4. Calculate partial gradient ΔW_i according to W_i and X set (input images set) for any worker e_i 5. Transmit the partially computed gradients' weights ΔW_i into the central repository server 6. End loop

Algorithm 2: ADSGD pseudocode in Master side

Inputs: learning rate alpha, Convolutional structure, Loss Function Lj and Workers E $\{e_1 \ldots\ldots e_n\}$ Outcome: global gradients' weights
1. Await reception of ΔW_i of all workers 2.$W_{i,j}$: process the partially computed gradients' weights ΔW_i Calculate ∇L_j; by Equation. (13) Calculate the globally aggregated gradients' weights W_{i+1} by Equation (11) 3.Updating W_i and storing it on the central repository server. 4.Transmit the globally aggregated gradients' weights to the workers

4 Simulation Results

The experimental outcomes of the suggested approach to malicious identification are introduced in this section. All the simulations are done on MATLAB 2024b. As outlined in Section 4.1, the analyzed malicious images, were collected from the Malimg database. During the evaluation of the given methodology, we used metrics like Accuracy, Precision, Recall, and F-score, which are described in Section 4.2.

4.1. Dataset

Malimg dataset was applied in this study. Freely available in the public domain on Kaggle, it is a malware classification benchmark with 9,435 executables belonging to 25 classes of malware. Malware

binaries were resized to 32×32pixel images through nearest-neighbor interpolation. Test images of three types of malware are shown in Figure 5.

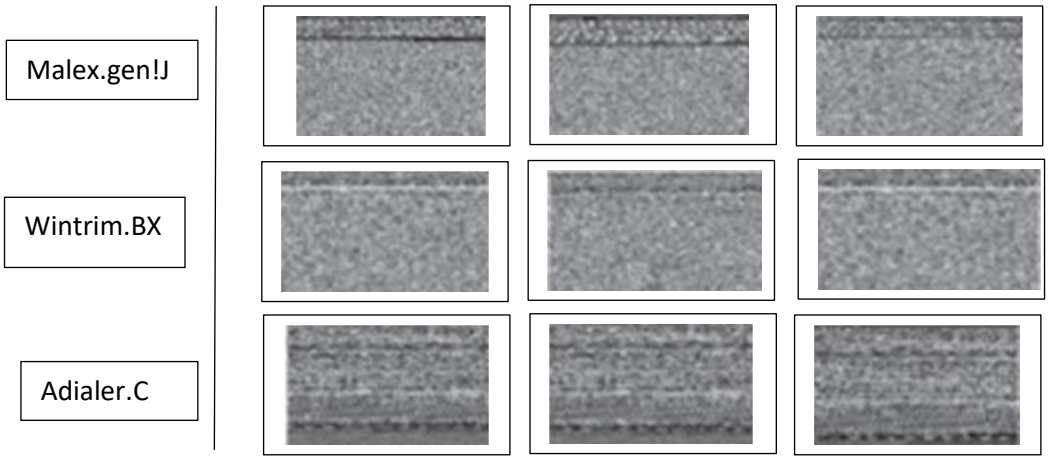

Figure 5: An instance for some kinds of Malicious Categories

4.2. Evaluation Metrics

Performance of the provided model was measured with the aid of parameters F-score, Precision, Recall and Accuracy that were calculated from the following formulas:

$$Acc = \frac{(TruePositive + TrueNegative)}{(TruePositive + FalsePositive + TrueNegative + FalseNegative)} \quad (14)$$

$$Precision = \frac{TruePositive}{(TruePositive + FalsePositive)} \quad (15)$$

$$Recall = \frac{TruePositive}{(TruePositive + FN)} \quad (16)$$

$$F1\ score = \frac{2 * (Recall * Precision)}{(Recall + Precision)} \quad (17)$$

In this case, $TruePositive$ is the count of true positive samples, $TrueNegative$ is the count of true negative samples, $FalsePositive$ is the count of false positive samples and $FalseNegative$ is the count of false negative samples.

4.3. Experimental based Result Evaluation

Figure 6 displays the Convolutional Network's learning process, highlighting its applicability in malicious content detection. The network's accuracy increases at an accelerating rate, whereby the high percentages (of approximately 95 percent to 100 percent) are achieved within less than 200 iterations by effective learning and best model optimization. The loss value, however, decreases in initial epochs and bursts off at very low points, showing optimal network optimization. These findings highlight the significance of accurate model parameters, fast convergence, and avoiding overfitting, all culminating in the network ability to detect malicious content.

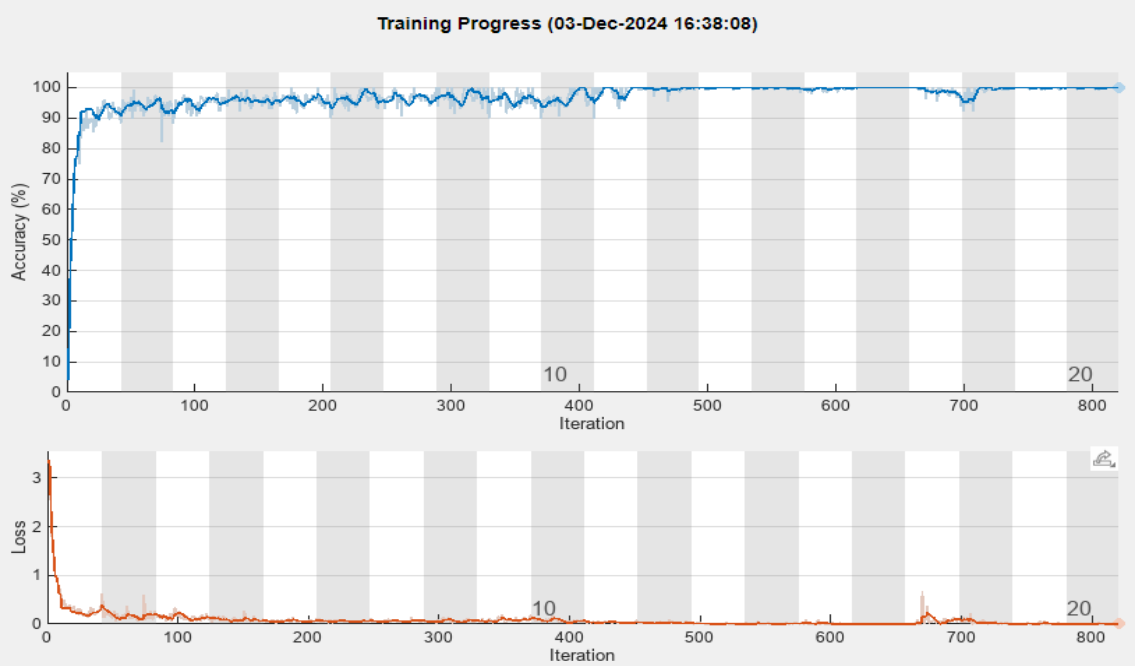

Figure 6: Convergence Curve of the Presented model

Confusion matrix of the classification outcome of the proposed approach is shown in Figure 6 From figure 6, it can be observed that distributed Convolutional Neural Network (CNN) has highly accurate and confident performance on test data. Value at the diagonal of the matrix is near 90 for all the classes and there is one point at 89 which appears to verify that the network is extremely precise in returning samples correct. These misclassifications are minimal and show the network's high ability to classify correctly between various classes of malware. The results show consistent behavior in all classes and confirm that the model can generalize very well and detecting malicious threats precisely.

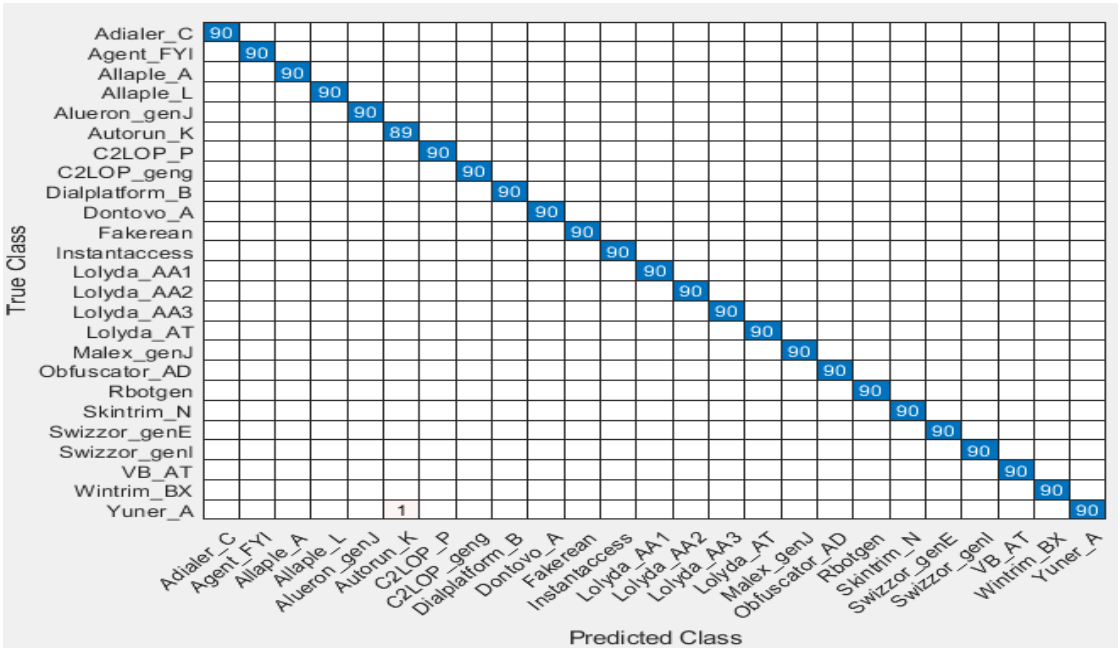

Figure 7: Confusion Matrix of the Model Provided

ROC curve is a measure of machine learning goodness of a binary classifier at various thresholds on the basis of two measures: true positive rate and false positive rate. The True Positive Rate (TPR), also referred to as sensitivity or recall, represents the proportion of actual positive instances that are correctly detected. On the other hand, False Positive Rate (FPR) is the ratio of the negative cases incorrectly classified as positive and is denoted by:

$$FPR = \frac{FP}{TN+FP} \tag{18}$$

The TPR/FPR trade-off can be represented as measure by plotting TPR against FPR for every threshold level. A good classifier is usually positioned close to the upper-left point of the ROC curve indicating high True Positive Rate (TPR) and low False Positive Rate (FPR). Conversely, a poor classifier is moved towards the lower-right hand corner which corresponds to a small TPR and large FPR. On the other hand, the random model class gives answers on the diagonal of the ROC plot with no discrimination as its true positive rate equals its false positive rate. Figure 8 shows the ROC plot of the proposed approach that is close to the top-left corner of the plot—showing its high true positive and low false positive rate. That way, it can be seen that the given model shows high potential in terms of effective malware classification.

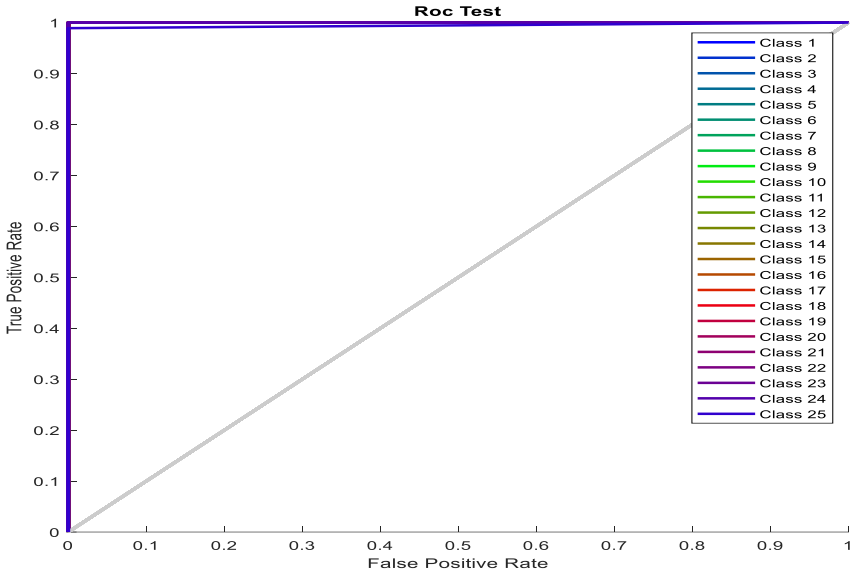

Figure 8: ROC Curve

The GWO-Distributed CNN method introduced here is found to be better than a number of state-of-the-art methods based on accuracy with the best accuracy being 99.95%. This is higher than the best-so-far result achieved by Al-Khater et al., who achieved 99.64% accuracy utilizing fast and adaptive bidirectional empirical mode decomposition and VGG-ResNet hybrid architecture. Similarly, Singh et al. and Shaukat et al. achieved accuracies of 99.51% and 99.30%, respectively, which are competitive but inferior to the proposed framework. Traditional CNN-based fusion methods also lag behind in predictive performance. The enhanced performance of the model results from incorporating the Grey Wolf Optimizer (GWO) in a proper way into a distributed convolutional neural network structure to improve feature learning and generalization capacity. The results validate the outstanding classification performance of the methodology and illustrate the benefit of implementing metaheuristic optimization strategies in deep learning models for difficult pattern discovery tasks.

Table 2. Comparative Analysis of Different Methods Based on Accuracy

Author	Algorithm	Accuracy
Al-Khater et al.	VGG-ResNet-FABEMD	99.64
Shaukat et al.	RegNetY320-OCSVM-FS	99.30
Singh et al.	GRU+CNN+CSBW-RF	99.51
Behera et al.	ResNet50-VGG16	99.28
Kumar et al.	ResNet50-CNN	98.50
Presented Method	GWO-Based-Distributed CNN	99.95

Fig. 9 compares the proposed malware detection technique—optimized parameterized distributed CNN—versus various other techniques tested on Precision, Recall, and F-score. It is evidently observable from the figure that the proposed technique performs superior to all other competing techniques on both the test metrics, showing its strong and stable performance across different domains of classification. These metrics are critical in evaluating the practical effectiveness of malware detection systems, as they collectively reflect the model's ability to correctly identify true malware instances (Recall), avoid false positives (Precision), and maintain a balanced performance across both aspects (F-score). The superior performance of the proposed method across these metrics highlights its strong generalization capability and its effectiveness in handling imbalanced or complex data distributions, which are common in real-world malware detection scenarios. The use of optimized parameters and a distributed CNN structure contributes significantly to enhancing feature representation and classification accuracy. Notably, among the existing methods, the VGG-ResNet-FABEMD approach ranks second in overall efficiency, yet it still falls short of the performance achieved by the proposed model. This further underscores the advantage of integrating metaheuristic optimization with deep learning in constructing more precise and reliable malware detection systems.

5 Conclusion

In this research work, the proposed malware detection model uses Distributed CNNs and the Grey Wolf Optimizer in a distributed computing framework. The suggested approch will be organized in two main stages. During the first stage, GWO algorithm, which is based on the hierarchical hunting mechanism of grey wolves, is used to tune the hyperparameters of CNN. This technique proves to be effective in balancing exploration and exploitation in large search spaces and offers optimum tuning of significant parameters learning rates, filter sizes and depth of the network. The two approaches are to employ a distributed architecture where several local CNNs are utilized as separate workers. Workers classify instances of malware and exchange parameters through a master server and form a distributed and scalable framework in malware classification. The two main issues that are addressed using the given model are the hyperparameter tuning and nature of malware classification problem. GWO is used to tune the hyperparameters of CNN in order to enhance detection ability and training rate of convergence to an increased degree. The distributed nature also enables easy elimination of the processing bottlenecks as well as the faults; hence, the system would still be operational even if part of the nodes or the network is under attack. The suggested model is scalable and can be utilized for extensive malware detection simulations. Experimental analyses on the benchmark Malimg dataset confirmed its effectiveness, with an impressive accuracy rate of 99.95%. This high accuracy is crucial in practical cybersecurity use-cases where identifying malware quickly and accurately is essential. The research results reveal the potential synergy between metaheuristic optimizers and deep learning models in solving real-world cybersecurity problems. The proposed framework presents a scalable, adaptive, and robust solution for fighting new

and advanced malware threats in real-world settings, providing a solid foundation for combating cyber threats.

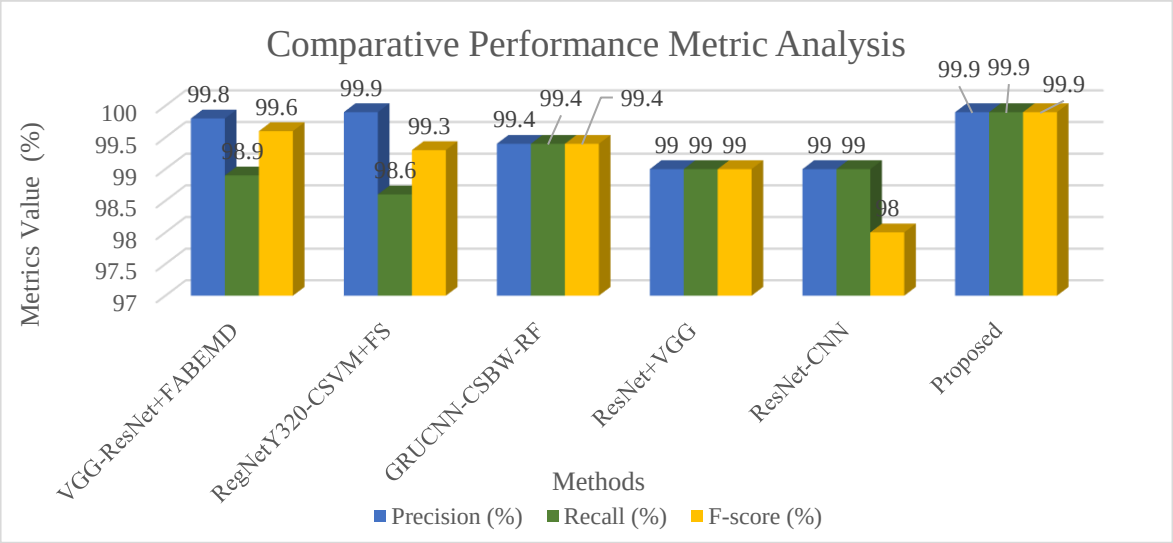

Figure 9: Performance Comparison of the Presented Approach Against Existing Models Based on Different Metrics

References

[1] Alavala, R., Vakalapudi, S., Ravula, V. M., Saisree, B., & Jukuntla, A. (2025, April). Signature-based Antivirus Scanner for Effective Malware Detection and Security Analysis. In *2025 5th International Conference on Trends in Material Science and Inventive Materials (ICTMIM)* (pp. 1047-1053). IEEE.

[2] Al-Khater, W., & Al-Madeed, S. (2024). Using 3D-VGG-16 and 3D-Resnet-18 deep learning models and FABEMD techniques in the detection of malware. *Alexandria Engineering Journal*, *89*, 39-52.

[3] Alzaben, N., Alashjaee, A. M., Maray, M., Alotaibi, S. D., Alharbi, A. A., & Sayed, A. (2025). Fortifying Android Security: Hyperparameter Tuned Deep Learning Approach for Robust Software Vulnerability Detection. *FRACTALS (fractals)*, *33*(02), 1-14.

[4] Armstrong, D., & Tanaka, Y. (2025). Boosting Telemedicine Healthcare Assessment Using Internet of Things and Artificial Intelligence for Transforming Alzheimer's Detection. *Global Journal of Medical Terminology Research and Informatics*, *3*(1), 8-14.

[5] Bakır, H. (2025). A new method for tuning the CNN pre-trained models as a feature extractor for malware detection. *Pattern Analysis and Applications*, *28*(1), 26.

[6] Behera, G. G., Pradhan, J., & Mishra, A. K. (2025). A Hybrid Deep Learning Malware Detection Model with The Fusion of VGG-16 and ResNet-50 Architectures. *Procedia Computer Science*, *258*, 1659-1668.

[7] Bhuiyan, S. M., Adhami, R. R., & Khan, J. F. (2008, March). A novel approach of fast and adaptive bidimensional empirical mode decomposition. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 1313-1316). IEEE.

[8] Dahiya, A., Singh, S., & Shrivastava, G. (2025). Android malware analysis and detection: A systematic review. *Expert Systems*, *42*(1), e13488.

[9] Ferdous, J., Islam, R., Mahboubi, A., & Islam, M. Z. (2025). A Survey on ML Techniques for Multi-Platform Malware Detection: Securing PC, Mobile Devices, IoT, and Cloud Environments. *Sensors (Basel, Switzerland)*, *25*(4), 1153.

[10] Gaber, M. G., Ahmed, M., & Janicke, H. (2024). Malware detection with artificial intelligence: A systematic literature review. *ACM Computing Surveys*, *56*(6), 1-33.
[11] Hasan, R., Biswas, B., Samiun, M., Saleh, M. A., Prabha, M., Akter, J., ... & Abdullah, M. (2025). Enhancing malware detection with feature selection and scaling techniques using machine learning models. Scientific Reports, 15(1), 9122.
[12] Hossain, M. A., & Islam, M. S. (2024). Enhanced detection of obfuscated malware in memory dumps: a machine learning approach for advanced cybersecurity. *Cybersecurity*, *7*(1), 16.
[13] Jiang, R., Weng, Z., Shi, L., Weng, E., Li, H., Wang, W., ... & Li, W. (2024). Intelligent botnet detection in IoT networks using parallel CNN-LSTM fusion. *Concurrency and Computation: Practice and Experience*, *36*(24), e8258.
[14] Ko, S. M., Yang, J., Kim, T., & You, I. (2025). N-gram opcode frequency based malware detection using CNN algorithm. *Soft Computing*, 1-9.
[15] Krishnan, D., & Singh, S. (2021, December). Cost-sensitive bootstrapped weighted random forest for DoS attack detection in wireless sensor networks. In *TENCON 2021-2021 IEEE Region 10 Conference (TENCON)* (pp. 375-380). IEEE.
[16] Kumar, S., Sapru, Y., Kumar, J., & Sapru, S. (2025). Image analysis and fine-tuned ResNet50 model for effective malware detection. *Procedia Computer Science*, *260*, 1098-1106.
[17] Liu, Y., Fan, H., Zhao, J., Zhang, J., & Yin, X. (2024). Efficient and generalized image-based CNN algorithm for multi-class malware detection. *IEEE Access*.
[18] Luedke, R. H., & Kingdone, G. (2025). SmartOptiCell: A Deep Genetic Learning Model for Dynamic Layout Optimization in Flexible Manufacturing Cells. *International Academic Journal of Science and Engineering*, *12*(2), 35-42. https://doi.org/10.71086/IAJSE/V12I2/IAJSE1216
[19] Mouhtadi, W. E., Bakkali, M. E., Maleh, Y., Mounir, S., & Ouazzane, K. (2024). Refining malware detection with enhanced machine learning algorithms using hyperparameter tuning. *International Journal of Critical Computer-Based Systems*, *11*(1-2), 48-67.
[20] Nandy, M., Dubey, A., & Verma, M. (2025). Detecting, bioaccumulating, and toxicological effects on aquatic trophic levels of nano plastic pollution. *International Journal of Aquatic Research and Environmental Studies*, *5*(1), 529–537. https://doi.org/10.70102/IJARES/V5I1/5-1-49
[21] Nawshin, F., Gad, R., Unal, D., Al-Ali, A. K., & Suganthan, P. N. (2024). Malware detection for mobile computing using secure and privacy-preserving machine learning approaches: A comprehensive survey. *Computers and Electrical Engineering*, *117*, 109233.
[22] Nguyen, P. S., Huy, T. N., Tuan, T. A., Trung, P. D., & Long, H. V. (2025). Hybrid feature extraction and integrated deep learning for cloud-based malware detection. *Computers & Security*, *150*, 104233.
[23] Perera, K., & Wickramasinghe, S. (2024). Design Optimization of Electromagnetic Emission Systems: A TRIZ-based Approach to Enhance Efficiency and Scalability. *Association Journal of Interdisciplinary Technics in Engineering Mechanics*, *2*(1), 31-35.
[24] Qi, X., Johar, M. G. M., Khatibi, A., Tham, J., Zhang, R., & Mao, S. (2023, December). Research on Malicious Software Detection Based on CNN-LSTM Hybrid Model. In *2023 5th International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)* (pp. 300-306). IEEE.
[25] Qureshi, S. U., He, J., Tunio, S., Zhu, N., Nazir, A., Wajahat, A., ... & Wadud, A. (2024). Systematic review of deep learning solutions for malware detection and forensic analysis in IoT. *Journal of King Saud University-Computer and Information Sciences*, *36*(8), 102164.
[26] Rathi, S., Mirajkar, O., Shukla, S., Deshmukh, L., & Dangare, L. (2024). Advancing crack detection using deep learning solutions for automated inspection of metallic surfaces. *Indian Journal of Information Sources and Services*, *14*(1), 93-100. https://doi.org/10.51983/ijiss-2024.14.1.4003

[27] Reddy, B. M. K., Abdul Azeez Khan, A., Javubar Sathick, K., & Arun Raj, L. (2025). Cyber Attack Recognition in an Internet of Things-Enabled Environment Using a Hybrid Optimised Deep Learning Approach. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 16*(1), 49-71. https://doi.org/10.58346/JOWUA.2025.I1.003
[28] Redhu, A., Choudhary, P., Srinivasan, K., & Das, T. K. (2024). Deep learning-powered malware detection in cyberspace: a contemporary review. *Frontiers in physics*, *12*, 1349463.
[29] Revathy, G., Suthanthiradevi, P., Sathishkumar, P., Sivakumar, S., Leo, L. M., & Rajasekar, A. (2025, May). Malware-Optimizing Neural Network Hyperparameters Using Swarm Intelligent Algorithm. In *Smart Grid Security and Protection: Proceedings of the International Conference on Sustainable Power and Energy Research, ICSPER 2024, Volume 2* (Vol. 1307, p. 45). Springer Nature.
[30] Shaukat, K., Luo, S., & Varadharajan, V. (2024). A novel machine learning approach for detecting first-time-appeared malware. *Engineering Applications of Artificial Intelligence*, *131*, 107801.
[31] Singh, S., Krishnan, D., Vazirani, V., Ravi, V., & Alsuhibany, S. A. (2024). Deep hybrid approach with sequential feature extraction and classification for robust malware detection. *Egyptian Informatics Journal*, *27*, 100539.
[32] Smmarwar, S. K., Gupta, G. P., & Kumar, S. (2024). Android malware detection and identification frameworks by leveraging the machine and deep learning techniques: A comprehensive review. *Telematics and Informatics Reports*, *14*, 100130.
[33] Wasif, M. S., Miah, M. P., Hossain, M. S., Alenazi, M. J., & Atiquzzaman, M. (2025). CNN-ViT synergy: An efficient Android malware detection approach through deep learning. *Computers and Electrical Engineering*, *123*, 110039.
[34] Zhang, S., Su, H., Liu, H., & Yang, W. (2025). MPDroid: A multimodal pre-training Android malware detection method with static and dynamic features. *Computers & Security*, *150*, 104262.

Authors Biography

Sadeq Thamer Hlama is Lecturer at the College of Science, Sumer University, Iraq /Thi-Qar Governorate /Al Rifaee. Major: Computer. His research focuses on big data, machine learning, and distributed computing systems.

Abolfazl Gandomi is associated with the Department of Computer Engineering at the Yazd Branch of Islamic Azad University in Yazd, Iran. His research focuses on Big Data, Machine Learning, and Distributed Computing Systems.

Zuhair Hussein Ali Department of Computer Science, College of Education, AL-Mustansiriyh University, Baghdad, Iraq. His Research focuses on artificial intelligence.

Mohammadreza SoltanAghaei is associated with the Department of Computer Engineering at the Azad Branch of Islamic Azad University in Isfahan, Iran**.** His research interests include Quantum (Algorithms, Computers, Networks and QKD-Quantum Key Distributed); Computer Networks (Internet of Things (IoT); Smart (Cities, Healthcare, Home, and Farming).