ENHANCED PARALLEL DEEP LEARNING FOR MALWARE DETECTION (EPDL-MD) MODEL

***Madihah Mohd Saudi*[1], *Chowdhury Sajadul Islam*[1,2], *and Nur Hafiza Zakaria*[1]**

[1]Cybersecurity and Systems Research Unit, Information and Security Assurance Programme, Faculty of Science and Technology, Universiti Sains Islam Malaysia, 71800 Nilai, Negeri Sembilan, Malaysia

[2]Department of Computer Science and Engineering, Daffodil International University, Dhaka 1230, Bangladesh

Emails: madihah@usim.edu.my*, chowdhury.uu@gmail.com, mzhafiza@usim.edu.my

ABSTRACT
The number of cyberattacks caused by malware targeting critical sectors, such as energy systems, telecommunications, healthcare, and finance, is rapidly increasing worldwide. The evolution of malware has made detection techniques more challenging, resulting in financial losses and reduced productivity. To address this issue, this paper presents the Enhancement of Parallel Deep Learning for Malware Detection (EPDL-MD) model. This model focuses on improving the parallel convolutional neural network (CNN) architecture. The performance of the CNN is influenced by its hyperparameters, and the enhancements have led to an increase in accuracy and learning rate. The experiment utilized 176,000 malware samples, which were sourced from 86 distinct malware families and one benign family. Based on the analysis and experiments, the EPDL-MD model has achieved an impressive accuracy rate of 99%.

Keywords: Malware detection; Malware attacks; Feature extraction; Deep learning; Convolutional neural network.

1.0 INTRODUCTION

Malware is harmful software commonly used to attack networks or machines. Trojan horses, worms, viruses, ransomware, adware, spyware, rootkits, keyloggers and cryptojacking are typical malware [1]. The threats posed by malicious software have grown significantly in computer systems. These dangers compromise systems at all levels, whether they belong to small businesses, big corporations, or even governmental organizations. With the help of advanced techniques, such as encrypting the harmful payload and altering the code structure at each infection to maintain the same functionality, their creation technology is continually growing. However, it cannot ensure complete security from the hands of hackers. The heterogeneity of modern malware prevents current technologies from adequately addressing this issue [2,3]. Hence, this paper suggested an Enhanced Parallel Deep Learning (EPDL) method for more accurate malware classification and detection using features of images of converted binary files. To assist in developing effective classification and detection models, key features were carefully constructed, and hyperparameters were tuned properly using the enhanced harmony search (EHS) algorithm.

Adjusting hyperparameters to fit the model's data, this paper experiments how various hyperparameters, such as the number of epochs, batch size, amount of layers and neurons, optimization technique, dropout rate, the type of activation function, and learning rate, affect malware detection through deep learning. A confusion matrix (CM) is used to validate the model, where the experiments show that EHS is more effective for determining the appropriate parameters, while the choice of hyperparameter numbers significantly impacts the accuracy of a model. This paper aims to explore various types of malware and their classifications while comparing traditional malware detection techniques with advanced methods like EPDL (Enhanced Pattern Detection Learning) about system vulnerabilities. In particular, the proposed method substantially increases the neural networks model's identification of malware accuracy on the proposed balanced dataset (99.0%) and imbalanced dataset (98.7%). These findings show how effective the proposed method is and have significant outcomes for the malware identification industry. The researchers applied a neural network, which works like a human brain. The filters are used as eyes to select the exact pixel to extract the natural features of malware. A convolution neural network (CNN) is an artificial neural network (ANN) that has many hidden layers that can automatically learn features and extract features and trained convolution neural networks (CNN) to distinguish benign files from malware files [4]. The research focuses on four solutions: practical deep learning tasks, creating a novel RGB dataset, and proposing a novel enhanced parallel deep learning framework and response software. It also focuses on tasks applied to security research topics [5]. The confusion matrix (CM) and k-fold cross-validation techniques evaluate the model's flexibility and generalization.

The paper is organized into several sections. Section 2 discusses the background of the paper. Section 3 explains the methodology for converting malware files into image files to aid in classification and detection methods. Section 4 presents the classification results and evaluates the model. Section 5 introduces the Malware Detection and Response System (MDRS). Finally, Section 6 offers conclusions and outlines the study's limitations.

2.0 BACKGROUND STUDY

The history of malware dates back to the 1970s, driven by technological advancements, increased connectivity, and the growing significance of digital information [1], [2]. It began with the experimental Creeper virus in the early 1970s and has since evolved into sophisticated attacks that continue to challenge digital security [1]. In Convolutional Neural Networks (CNNs), such as LeNet-5 [6], shift-invariance is achieved through subsampling layers. Neurons in these layers receive input from a small, non-overlapping receptive field of the previous layer. After each neuron computes the total of its inputs, a non-linear transfer function is applied in the algorithm[7]. This process involves multiplying the total by a trainable coefficient, adding a trainable bias, and multiplying the result by another [8].

Different attack detection methods, such as the Hidden Markov Model (HMM) and Support Vector Machine (SVM), have been proposed in previous studies [9]. The author's proposed model follows the creation target by using a Deep Q-network (DQN) to send data based on the waiting duration of every router threshold. A fully linked feed-forward (FF) multilayer perceptron (MLP) model is applied to the neural network (NN) as the mean opinion score (MOS) for secure encrypted communication over a 5G network with less bit error rate(BER) [10]. According to the National Computer Network Incident Response Technical Team/Coordination Center of China (CONCERT/CC), over 96,200 hosts in China were infected by network viruses, and malicious programs were transferred as many as 69.724 million times in just one week [11].

Song *et al.* [12] implement a dual super-resolution CNN (DSRCNN) to generate high-quality pictures. The DSRCNN employs two sub-networks to obtain complementing low-frequency features that boost the super-resolution (SR) network's learning ability. Dual sub-networks are used in conjunction with a convolutional and residual learning process to avoid long-term reliance. An improved block captures original information and obtains high-frequency information from a deeper layer using sub-pixel convolutions, eliminating information loss in the original picture. A feature learning block is used to learn more details on the high-frequency data to get more high-frequency features. The recommended approach is best suited for complex scenarios requiring good image quality [13] and proposes an enhanced, faster dual CNN, R-CNN traffic sign detection technique. The ResNet50-D feature extractor, attention-guided context feature pyramid network (ACFPN), and Auto Augment technique produce the quickest R-CNN model. Mayuranathan *et al.* built a features-based intrusion detection system employing the RBM model by assessing a unique compound picture descriptor, while the author B. Sun [14] has been exploring an automated and online crowd anomaly recognition approach. A dual-channel convolutional neural network (DCCNN) has been configured to efficiently analyze scene-related and motion-related crowd data derived from raw frames and compounding descriptors instances. This study uses a dual-stream convolutional neural network (CNN) based on sEMG modeling analysis to conduct feature fusion of sEMG energy kernel phase portrait and inertial measurement unit (IMU) data for gesture recognition. Gibert *et al.* paper makes several contributions, including an extensive description of methods and ingredients in a conventional machine learning workflow for malware classification and identification, an analysis of current trends and advancements in the field, focusing on deep learning methods, and an analysis of future research direction. The Viola-Jones face detection technique is utilized in this work to identify the face and extract the eye region from photos of faces by Chirra *et al.* A stacked deep convolution neural network is created for the learning phase to extract features from dynamically determined important frames from camera sequences. The CNN classifier uses a SoftMax layer to categorize the driver as asleep or awake. When the driver feels sleepy, this system informs them with an alarm. Meanwhile, dual network models employ two separate CNNs (CNN-L and CNN-R) to train stereo pairs' left and right pictures [15].

Additionally, it has been found that, in most instances, the DNM12 model performs better than the DNM6 model. Li *et al.* suggest a dual CNN RE model based on a knowledge-based attention mechanism that employs word embedding and entity embedding from a knowledge base as CNN input [13]. The attention mechanism allows entity embedding to get more sentence characteristics, while word embedding can obtain more background information to anticipate relations. A framework based on dual convolutional neural networks that were created to train the global and attention representations of the lncRNA-disease connections is used in their application. The framework's right portion learns associations' attention representation, while the left component uses the various information in the feature matrix to understand the global representation of lncRNA-disease relationships. A dual convolutional neural network (CNN) architecture is presented by [16] for automatic anomaly identification in complicated security imaging. Researchers have developed object localization variations for certain object classes of interest by utilizing current developments in region-based (R-CNN), mask-based CNN (Mask R-CNN), and identification framework like RetinaNet. Researchers then construct object anomaly detection as a two-class problem by using a variety of known CNN objects and fine-grained category classification algorithms (anomalous or benign). A low-cost blind spot detection system based on a lite object detection algorithm and limited resources hardware are used by the author, which helps with the features of malware for this research [17]. CornerNet-Lite combines CornerNet-Saccade, which employs an attention mechanism to do away with the need to process every image pixel exhaustively, and CornerNet-Squeeze, which adds a new compact backbone architecture. Together, these two variations meet the crucial use cases for ineffective object detection: enhancing real-time efficiency without losing accuracy. It is observed in the research [18] that RGB images work better than grayscale images in deep learning when tuning the hyperparameters. Then, the researcher planned to use RGB with a large dataset with parallel deep learning with the EHS algorithm, which performs better than other models. The increasing volume and variety of malware significantly hinder network security, making developing rapid and accurate methods to identify and categorize malware and its variants a top priority in the professional landscape.

The findings presented in this review are based on prior research related to malware and deep learning, highlighting improvements based on identified gaps in existing literature. Additionally, using a parallel-deep learning model for threat detection remains underexplored, with only a few studies available. These findings are further detailed in Section 3, which focuses on methodology.

3.0 METHODOLOGY

The research methodology includes collecting and analyzing data to feed the EPDL-MD models, which are designed to detect malware accurately and respond to it. The results are then examined in the context of recent research on malware identification techniques. To categorize and detect malware-based cyber attacks, the malware binary files are first converted into images before using the proposed technique CNN models [19, 20]. The enhanced parallel convolutional neural network (CNN) algorithm is used as EPDL to identify and classify malware families from the converted image dataset. By doing this, researchers learn about the many malware variants and their behaviors. To achieve the ultimate goal, the researchers enumerated the proposed CNN models. In the next section, researchers outline the techniques for this classification and detection procedure.

Fig. 1 shows the flow diagram of the research methodology, where the malware detection framework begins with binary malware datasets downloaded from various sites and Python scripts are loaded to transform RGB images. In the proposed method, input data is split for training, validation, and testing purposes to fulfill the backpropagation mechanism of the deep learning algorithm. After tuning the hyperparameter to fit the model, it went to the performance validation phase. There was a dropout layer between fully connected (FC) layers to tune the number of parameters using the EHS algorithm. A more FC layer in parallel CNN can achieve better results when an ensemble of all CNNs uses concatenate techniques. The researchers used duel-CNN as an example of parallel CNN.

The validation set is used during the deep learning model's training phase to evaluate its performance in each epoch and fine-tune its parameters—training output per epoch validated by the validation data set. If the model becomes overfit or underfit, the researchers cannot finalize the model; even if the validation is not good, it is not finalized, and then it can be checked by others' value of parameters and start backtrack tuning from the beginning. After finalizing the model, the testing data will be used to test the model and calculate the performance using a confusion matrix (CM) and 10-fold cross-validation. At the same time, the finalized model is saved, and it will go to the multiclass classifier to detect the malware class or benign class. This model is saved as a *.pt* extension of PyTorch for the development of response software by using the Python TK framework. The entire process is illustrated in Fig. 1.

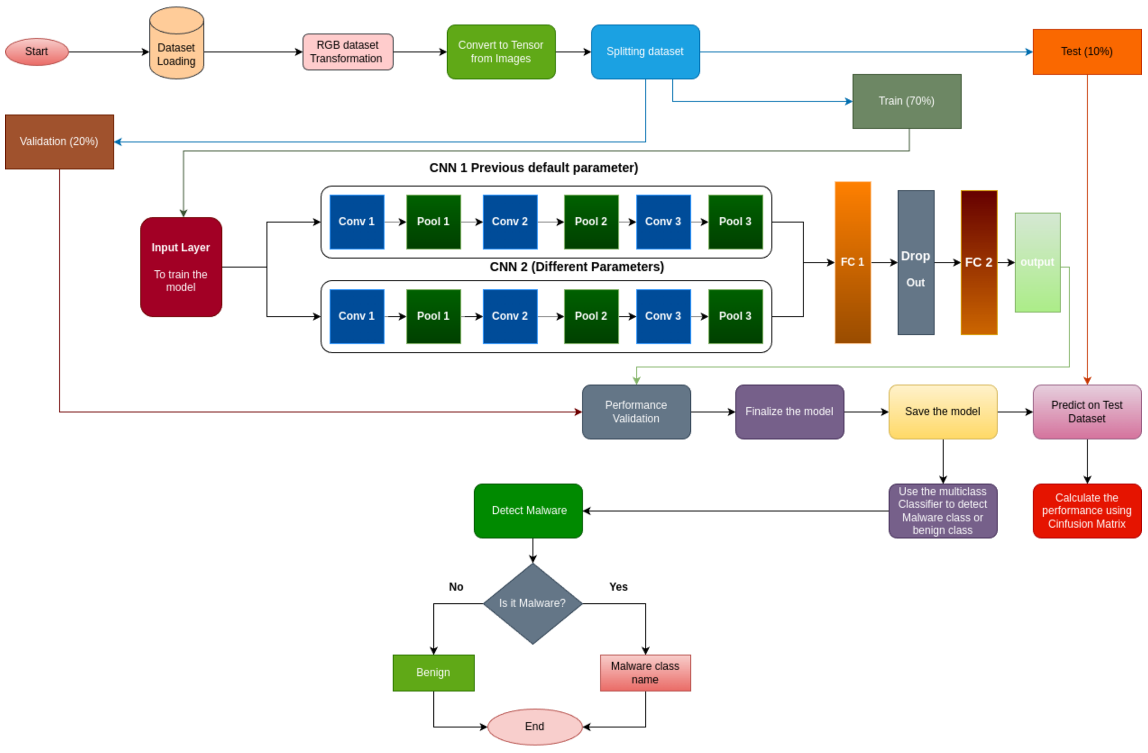


Fig. 1: The flow diagram of the proposed malware detection approach

3.1 Malware Data Acquisition to Create Dataset

The researchers made the dataset available for exploratory use after collecting it from VX-Underground between August 2020 and October 2024. This dataset consists of 3-channel images derived from Portable Executable (PE) files, representing both malicious and benign samples, as illustrated in Fig. 2, which depicts the image processing system. The researchers downloaded binary malware files (.exe, .apk, .ipa, .ipsw, .bin, .zsh) totaling 24.7 GB from 86 different malware families. In total, the dataset includes 186,000 malware samples from these 86 distinct families, as well as one benign family. To create this dataset, the researchers used image-based RGB files obtained from sources such as Vx-Underground, Malevis, VirusTotal, and Malware Bazaar, converting a variety of recently discovered PE files. After downloading the malware samples, the researchers unzipped and cleaned the data before compiling the dataset of PE files. The benign samples were collected from PortableApps and Softonic. Links to the malicious websites are provided in Table 1.

Table 1: List of the data set collection websites

Serial No.	Data set	Links
1	Malevis	https://web.cs.hacettepe.edu.tr/selman/malevis/
2	Malware Bazaar	https://bazaar.abuse.ch/browse/
3	Virus total	https://www.virustotal.com
4	VX-undergroun d	https://samples.vx-underground.org/samples/Families
5	Total Virus	https://www. totalvirus.com

3.2 Data Preprocessing and Organization

The initial phase of the methodology involves several crucial preprocessing steps for malware images, which are illustrated below.

3.2.1 Dataset Loading and Transformation

The raw binary dataset is loaded and converted into RGB image format, as shown in Fig. 2. Input images are converted into neuron/tensor format to feed the input of the EPDL model for compatibility with deep learning frameworks. Data normalization is applied to ensure consistent input scaling. Data augmentation is also applied to balance datasets to avoid overfitting.

3.2.2 Dataset Splitting

The dataset is divided into two distinct sets:

Training Set (70%):	Used to train the model
Test Set (30%) :	Reserved for final performance evaluation

3.2.3 Methods of Binary-to-Image (B2I) Conversion

The researchers converted binary files into image files because an image dataset is necessary to implement CNN-based deep learning techniques for the convolutional layer input. This effective conversion method results in an RGB image where the pixel values range from 0 to 255, with 0 representing black and 255 representing white. The detailed procedure is outlined and illustrated in Fig. 2, and converted image samples are displayed in Fig. 3. Using the PE-to-image conversion technique, a malicious binary is transformed into a vector of 8-bit unsigned integers arranged into a two-dimensional array. Finally, a color map is added to the 2D array to visualize the malware binary effectively.

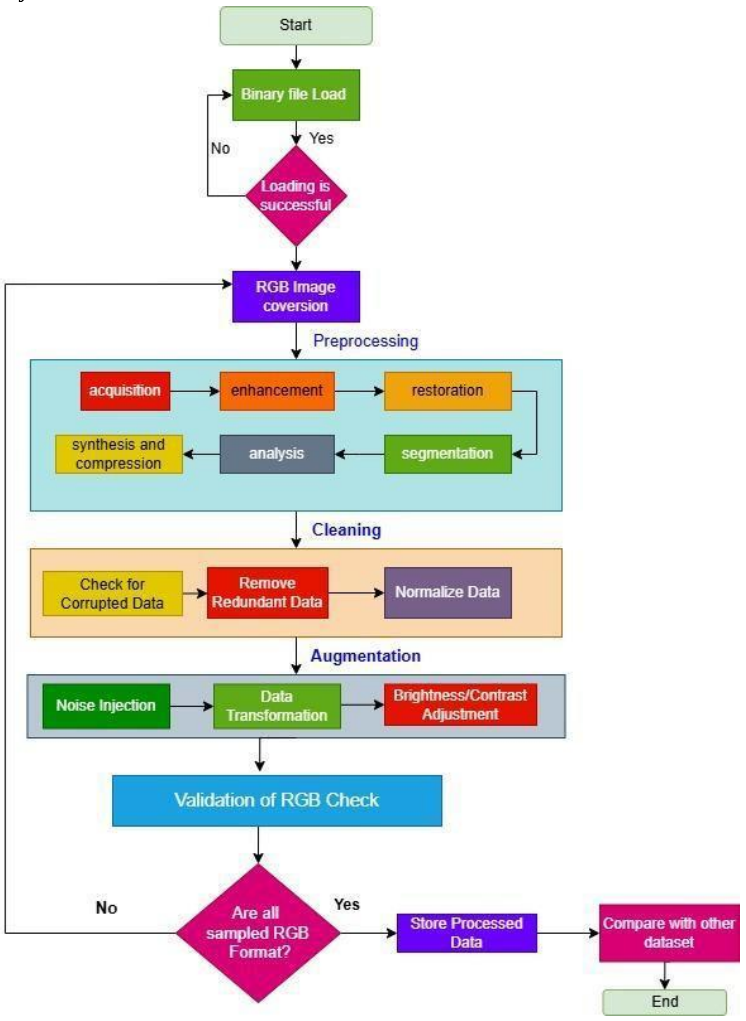


Fig. 2: A process flow diagram of malware binaries into RGB images conversion

After converting the malware, the following samples of the image are in Fig. 3

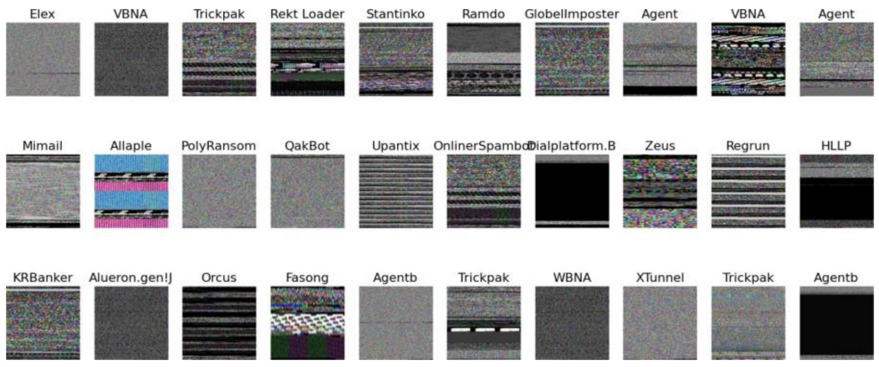


Fig. 3: Binary malware to RGB image conversion samples from 86 malware families.

The binary filename extensions are .bin, .dll, .exe, .apk, etc., and are examples of applications commonly known as PE files. The researchers collected malware PE files of various classes between April 2022 and September 2024. The researchers used the open-source websites VX-underground, VirusShare, Malevis, and MalwareBazaar to obtain all the relevant malware data. For the benign or non-malicious PE files, the researchers collected benign data from licensed operating systems and software sites like SourceForge and CNET, two open-source community resources. These PE malware and benign files (.exe, .apk, .ipa, ipsw, .bin, .zsh) are converted into image files using Python for deep learning.

The augmentation pipeline generated architecture for analyzing images, allowing for sequential processing that uses less memory. By creating augmented samples dynamically while training, this method avoids buffer overflow problems, which makes it especially useful for massive datasets. Fig. 4 shows the output of image augmentation. The phases of the data flow process are given in the following:

- A source folder is used for loading the initial pictures.
- Pictures are scaled to the required 224 × 224 pixels.
- There is a 50% chance of applying horizontal flipping augmentation.
- Particular identification codes are preserved with augmented photos.

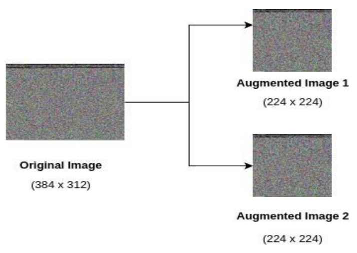


Fig. 4: Image Augmentation to balance the dataset

3.2.4 Normalization

It is the pre-processing of data before it is given to the neural network. Inspired by the research on lower-quality camera lens phone exactness with conventional high-end light microscopes, the researcher uses a pixel classifier framework in Convolutional Neural Networks (CNN) to classify the object classifier area of malaria in red blood cells or white blood cells. The system suggests that specialists reduce the workload and increase the accuracy of the decision without human intercession, or, as a guide, it permits RGB 3-channel feature detection of malware in this research [21]. They are difficult to classify since they lack specified dimensions and are produced by PE files. This problem was initially fixed by scaling down the images to 224 x 224 pixels while converting. The conversion code included modules like NumPy, image.io, and PIL, were included in the conversion code. Using NumPy's reshape function, the researcher normalized the data by making all of the photos 224 x 224 in size. Feeding the dataset into a DL model requires normalization for various dimensions produced from PE files, which are challenging to categorize. The images were re-sized to 224×224 pixels to solve this problem.

After image processing, a minimum of 177, a mean of 1953, a median of 844, and a maximum of 10,000 samples are included in the dataset. To mitigate the risk of overfitting and to enhance the classification process, this research adopted the percentage (70-30) split approach, also known as hold-out validation [22]. The 10-fold cross-validation is also applied as a comparison. While the k-fold technique repeats the process k times, the percentage split is executed only once. A common distribution is to allocate 30% of the data for testing and 70% for training [23]. Based on pilot testing, the 70-30 split produced superior results compared to both the 80/20 split. Further details regarding the dataset utilizing this percentage division method are shown in Table 2.

Table 2: Dataset distribution

Dataset	**Training**	**Testing**	**Total**
Malware	123,200	52,800	176,000
Benign	7000	3000	10000

Finally a 10-fold cross validation used to validate the whole model and dataset. In this paper, the researchers mentioned only 86 families of malware. The following Fig. 5 shows the unbalanced samples for each malware family from the 86 families and 1 benign family sequentially.

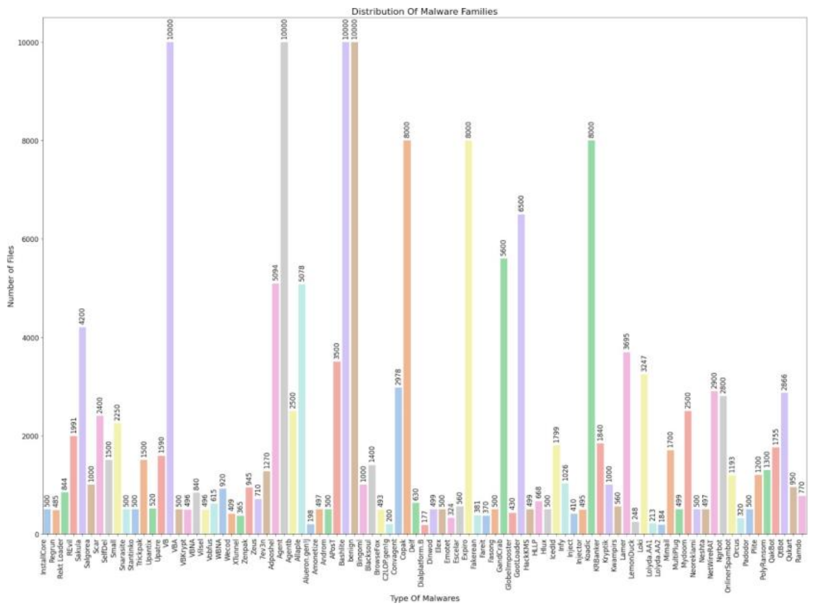


Fig. 5: Dataset distribution of samples for malware from the 86 malware families and 1 benign

It is shown in Fig. 6 that the height and width of the image size also increase with the ratio of storage size.

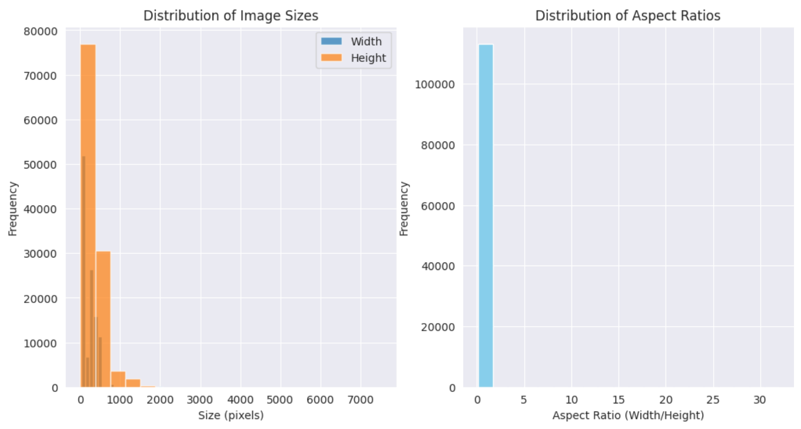


Fig. 6: Comparison of image size and ratio distribution of 86 families of malware and 1 benign family

3.3 Parallal Convolution Layer Configuration

Two convolution channels are included in the parallel CNN model, corresponding to the pooling and convolution layers, respectively. As a result, there will be no interference in the pooling and convolution calculations [23]. The two CNNs may train and generate the whole model using image data at the same time without any mutual influence. As referred to in Fig. 1, which illustrates a graphic representation of the proposed parallel-channel CNN model. The layers on the parallel CNN architecture exactly are not the same as those on a single CNN. This section describes the layers utilized in our parallel CNN design in-depth.

In the parallel convolutional layer, one matrix contains a set of learnable parameters known as the kernel. The other matrix acts as the restricted, and the output is the dot product of these matrices. The kernel generates dimensions proportional to the image's height and width. This two-dimensional representation, known as an activation or feature map, offers the kernel response for each spatial point in the picture, and the sliding size is known as the kernel stride [24]. The filter specifies the number of convolutions the researcher wants to generate. The researcher used 48 filters for the first convolution layer, 54 for the second convolution layer, and 57 for the final convolution layer in our parallel CNN-1 model. The number of 4 convolutions is specified by the filter and 3 x 3 kernel sizes. The researchers used 93 filters for the first convolution layer, 52 filters for the second convolution layer, and 172 filters for the final convolution layer. The 3 x 3 kernel sizes in the parallel CNN-2 model are shown in Table 3.

Table 3: CNN model parameters configuration.

CNN-parameters	**Shape**	**CNN1-Values**	**CNN2-Values**
Conv2d	87× 2,046	1344	2604
Batch Normalization	–	224	224
Max Pooling 2D	–	128	256
Conv2D	87×3843	256	512
Batch Normalization	–	64	128
Max Pooling 2D	–	64	92
Linear layer	–	512	256
Dropout	–	0.44	0.5
Activation Function	–	16 (ReLU)	24 (ReLU)
Rate of Learning	–	0.0001	0.0001
Decay rate	–	0.001	0.001
Optimizer used	–	Adam	Adam
Batch size	–	64	64
Epoch rate	–	15	20
Loss factor	–	Cross entropy	Cross entropy
Activation		Soft-max	Soft-max
Metrics	–	Validation Accuracy	Validation Accuracy
Fully Connected	–	512	256
Fully Connected	–	512	256
Output Layer	87×1	87	87

The full layer is used to convert the data into a 1D array as input for the FC layer [25]. The result of flattening the output of the previous layers is a single long feature vector. In this layer, FC-1 is used for 512, and FC-2 is used for 256 of the 1D array.

3.3.1 Tuning the Model

A horizontal flip augmentation approach was employed to create a balanced image dataset because the sampled dataset contains a lot of unbalanced malware and benign images. Therefore, using validation and testing techniques during training minimized over-fitting as much as possible. The suggested EPDL offers the most impressive and superior detection for malware and benign photos compared to existing ANN or CNN algorithms. Therefore, this paper examines the architecture, training behavior, fine-tuning and optimization hyper-parameters, and attainment of detection outcomes. All convolutional and max pooling layers use ReLU activation with a stride of 1 and a kernel size of 3×3. Since its layered sections have intrinsic repeating different sorts of layers that are easy to update and adjust, the proposed fine-tuned CNN model for RGB datasets has the most substantial advantage over default CNN algorithms: it may be improved. Additionally, with only a few training repetitions, the presented refined algorithm improves detection performance without requiring more in-depth training. The consisting convolution or separable convolution layers use separate kernels to recognize and extract distinct texture-like characteristics in benign and malware images. It is, hence, very efficient in computation and appealing for use in identifying PE malware attacks. Along with making use of the advantages of the presented DL model, all of the hyperparameters of the employed CNN algorithms were adjusted. This fine-tuning works better than inexpensive processing complexity vs. fine-tuning and accurate detection versus shallow tuning. Therefore, every hyperparameter used in CNN algorithms is adjusted using the DL technique until a high and efficient identification rate is attained. The following are the final fine-tuning and optimizing variables used in the proposed vision-based DL model: Adam optimizer, linear regression regularization (L2-regularization) with a weight decline rate of 0.001 (primarily user assigned finally, HS fixed it), a learning rate of 0.001, a dropout rate of 0.5, a minimal batch size of 32, and a maximum of 15 epochs, with learn speed scheduling variables set to "piecewise" learn rates drop precedence to maximize the efficiency of the training and validation process and avoid over-fitting, the hyperparameters have been carefully chosen. Furthermore, in order to improve the hyperparameters and weights associated with the various layers in the proposed CNN model—which was first created to achieve greater identification efficacy for identifying malware attacks—the designed DL model also uses the back-propagation technique.

3.3.2 Proposed Enhanced HS (EHS) Model to Tune Hyper Parameter

One of the key factors affecting an algorithm's effectiveness is parameter tuning, which is impacted by optimization issues. Pitch adjustment rate (PAR) and step factor bandwidth (BW), two critical parameters of responsive algorithm efficiency, are established using a dynamic adjusting method for parameters, which the IHS inspires algorithm [26]. To swiftly adjust to the present optimization challenge, their parameters are dynamically changed based on the number of iterations. By using how the parameters stored in the HM are formed, the EPDL-MD algorithm automatically calculates the HMCR and PAR when creating the subsequent harmonies. Using the HMCR and PAR set at the start, the generic HS algorithm is used repeatedly to update the HM in the first step of the EPDL-MD method. The HM value is used to calculate HMCR and PAR [27]. The approximate times the pitch adjustment mechanism and the harmony memory consideration technique are utilized in the EPDL-MD algorithm to calculate the HMCR and PAR.

3.3.3 *Techniques for Tuning* Hyperparameters of EPDL Using an *EHS* Algorithm

Finding the optimal hyperparameters is a problem in hyperparameter optimization for machine learning algorithms. Techniques for optimizing hyperparameters include grid searches, search at random, Bayesian optimization, and gradient-based optimization [28]. According to Thornton *et al.*, auto-weka is a classification method that combines hyperparameter optimization and selection. CNN's hyperparameters must be configured to include parameters like kernel size, stride, number of channels, and zero-padding, as well as the ANN's hyperparameters [29].

This paper proposes the EPDL-MD methodology to modify the hyperparameters in the CNN extraction of the scene's features. The suggested method for modifying the CNN feature extraction process's hyperparameters that affect the layer's feature map length is as follows. In the EHS algorithm, every harmony or solution is stored in the HM as a vector. Individual harmony is produced through random choices, harmony memory consideration, and pitch alteration approaches.

The proposed approach establishes the initial HMCR, PAR, and harmony memory length (HML). Once the EPDL-MD algorithm's parameters have been established, decide which operation should be carried out at each CNN layer and set the CNN's hyperparameters to harmony. Harmony vectors are created by expressing harmonies as vectors and utilizing the random selection approach. The CNN's loss for every harmonic vector is computed to make an initial HM. This procedure is carried out as frequently as the HMS procedure, which is carried out as frequently as HMS to create an initial HM made up of losses and harmony vectors. A new harmony vector is produced following the creation of the original HM and is used to compute the loss. Just utilizing a training dataset to train the CNN yields the loss.

3.3.4 Pseudo-code of EHS of EPDL-MD Framework

The stages of the EPDL-MD algorithm mirror those of the HS method, with an emphasis on performance enhancement. The following clearly demonstrates how the EPDL-MD algorithm functions:

Phase i: Configure the algorithm and the optimized solution with the necessary inputs.
Phase ii: Set up the harmony memory (HM) library and make assumptions about the network model's quality. The value of fitness is the training times T_t produced in the test set used by Equation (1)
Phase iii: Use the dynamical tuning techniques of parameters to update the pitch modification likelihood PAR and phase factor BW and assess whether a fine modification is necessarily used by Equation (2).
Phase iv: The self-decision search technique based on the best phase determines the harmonic cycle if fine-tuning is necessary; if not, the New Harmony is randomly introduced from solution space.
Phase v: Determine whether to retain the New Harmony based on the regional rivalry update plan.
Phase vi: Determine if the existing creation times exceed this maximum number and decide whether to end. Repeat Steps 3–5 until the maximum creation times are achieved.

Fig. 7 displays the EPDL-MD algorithm's general flow chart, where EHS is initialized and checks if it achieves the expected result. The process and local harmony are finished if the best result of global harmony is achieved. It cannot fit the best HM if it does not reach the expected global and local optima result. It starts execution from the beginning to update the harmony initialization process of EHS and then sets first the global best result and then the local best result.

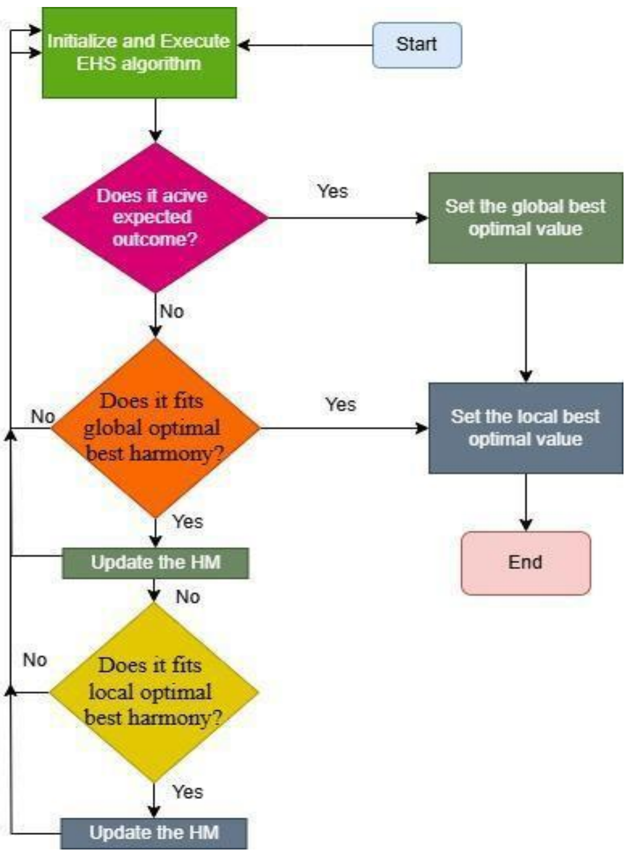

Initialize and Execute EHS algorithm
Start
Does it acive expected outcome?
Yes
Set the global best optimal value
No
Does it fits global optimal best harmony?
No
Yes
Set the local best optimal value
Yes
Update the HM
End
No
No
Does it fits local optimal best harmony?
Yes
Update the HM

Fig. 7: The data flow of the EPDL-MD to tune the hyperparameter using the EHS

3.3.5 Parallel CNN's mathematical model used in Harmony search

This section lays out the combination of CNN and enhanced harmony search (EHS) algorithm used to optimize the hyperparameter to get the effective result in this research; below is the flow diagram shown in Fig. 8. It aims to create a parallel CNN structure and an effective approach for automatically tuning the hyperparameters for a particular image classification task [29].

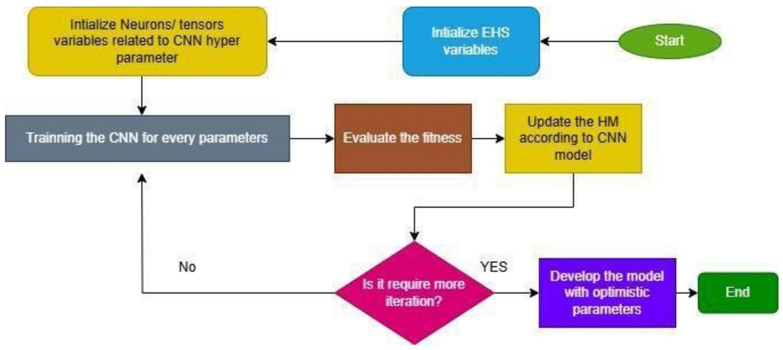

Start
Intialize EHS variables
Intialize Neurons/ tensors variables related to CNN hyper parameter
Trainning the CNN for every parameters
Evaluate the fitness
Update the HM according to CNN model
Is it require more iteration?
No
YES
Develop the model with optimistic parameters
End

Fig. 8: Enhanced HS Algorithm with CNN flow diagram for the proposed techniques

Since it would require a substantial amount of time and computer resources to optimize each hyperparameter of parallel CNN, the following ones were adjusted in the proposed research for this paper. The number of FC layers, convolution layers size and amount of filters, the hidden units in the FC layer, and the layer of dropouts are dynamically adjusted by the EHS algorithm.

The enhanced Harmony Search algorithm incorporates several key developments: a local competition update strategy, a dynamic selection search strategy based on the current state, and a dynamic parameter adjustment approach. These advancements effectively tackle the complex optimization problems that arise in large search spaces. Additionally, researchers have developed a new evaluation function that reduces computational costs without compromising the effectiveness of the search. It conducted comparative tests with the Malvis, MSBig15 dataset, and a proposed dataset to demonstrate the superiority of the Enhanced Harmony Search (EHS) algorithm introduced in this research. The EHS algorithm will serve as a hyperparameter optimization technique and explore how hyperparameters impact the fitting behavior of the network model. This analysis aims to produce more efficient optimization function values, ultimately saving time and computational resources. EPDL uses a confusion matrix and validation methods to assess the model's performance, comparing the proposed model against six other well-known deep learning models.

4.0 Experimental Results and Discussion

This section evaluates the EPDL model malware detection and the experimental results of the proposed model's performance. This model uses image pixel detection and pattern-matching algorithms to differentiate between malware and benign software. The proposed technique and dataset are evaluated not only by confusion matrix and cross-validation but also by comparing with six different DL classifiers and two different datasets. When the findings from the proposed model were compared to earlier studies of other researchers, it was found that the proposed model outperformed in accuracy and prediction rate compared with others. The experiment uses the total number of input neurons or tensors in the dense layer, the dimension, the variety of kernels for each layer, and the depth of the convolutional layer of the parallel CNNs as optimization hyperparameters. According to the experimental results, the proposed method improved the current technique and offered an average detection rate of accuracy 99.00%. The malware binary data was collected from the five sites from August 2020 to October 2024 for experimental purposes in this research. The dataset consists of binary representations of Windows/Android PEs, with consents having malware and benign samples. Preprocess the image data by removing duplicates; darks are shown in Fig. 2. The process used in Fig. 2 is image conversion, data balancing and normalization, data splitting, labeling, and handling missing values. The mathematical model used in the evaluation is shown in Fig. 9, where an image is taken as input, the pixel value is used as a feature, a new pattern of images is generated after ensembling the model, and the pattern is matched with the previous pattern to detect the malware.

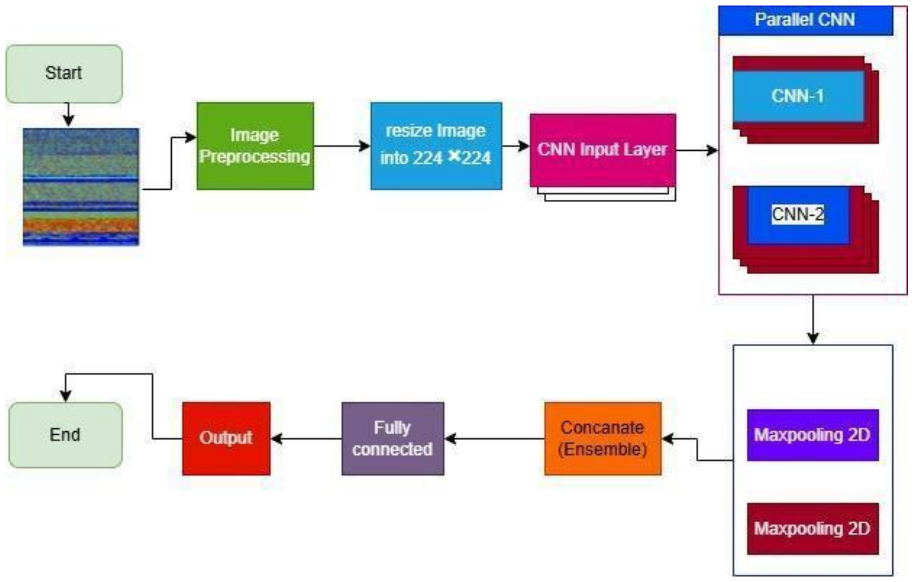


Fig. 9: Neurons/ Tensors processing steps of the proposed model integrated with hidden EHS algorithm.

The libraries and additional packages are usually installed instantly during the installation of Python 3.11 to set up the experimental environment. Anaconda Python framework is used for all tests in this paper. The majority of the malware classes are present in this collection. The novel dataset contains 86 families of malware and one benign family, with 186000 malware samples in the balance dataset and 179889 malware samples in the imbalance dataset. This can be fixed by including experimental material. The software and hardware setup used for the experiment is listed in Table 4.

Table 4: Hardware and software requirements for training and testing.

Hardware	**PC Configuration**	
	Processor:	Intel Core i5 13 Gen
	GPU : Memory : Storage :	GeForce RTX 3060, Nvidia's GA106 with 3840 unlocked CUDA cores RAM 32GB NVMe SSD
Software	Kali Linux, Python 3.11 programming with libraries including PyTorch framework, Tensor Flow framework, Keras, Numpy, Matplotlib and Sykit Learn, scikit-learn frameworks and libraries	
Datasets	Kaggle, Malimg, Virusshare, Malvis, MS Big2015, VX-underground	

The investigation demonstrates that the training times (T_{ts}) for the network model fitting decrease with the increasing learning rate (R_L). The T_{ts} of the network model needed to fit increases with decreasing learning rate (R_L). The larger the batch size (β), the more T_{ts} the model of the network needs to be fitted; the smaller the batch size, the less time the network model needs to be trained. In the binary categorization process, both malicious and benign files are used in this detection to observe how it turned out. The new data set accuracy and loss performance compared with the various models are shown in Table 5 below.

Table 5: Experimental Results for testing dataset the RGB dataset.

Malware	**Precision**		**Recall**		**F1- score**	
	Balanced	Imbalance	Balanced	Imbalance	Balanced	Imbalance
Accuracy	0.99	.98	0.99	0.98	0.99	0.98
Weighted avg	0.99	.98	0.99	0.98	0.99	0.98

The following Fig. 10 shows the hyperparameter tuning of the balanced dataset, where the testing and validation accuracy are very high, and the loss level is near zero.

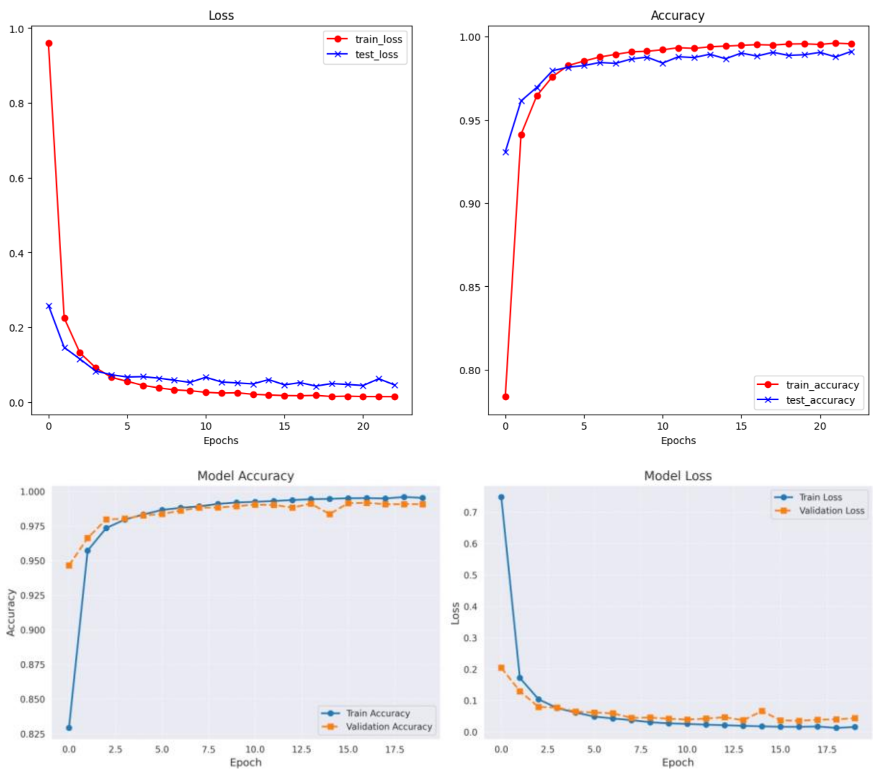


Fig. 10: The training, test and validation accuracy and loss for multiclass RGB balanced dataset of 86 malware families

The following Fig. 11 shows the confusion matrix (CM) of the hyperparameter tuning of the balanced dataset, where the accuracy is better and loss are lower than the imbalanced dataset of Fig. 12 compared with Fig. 10.

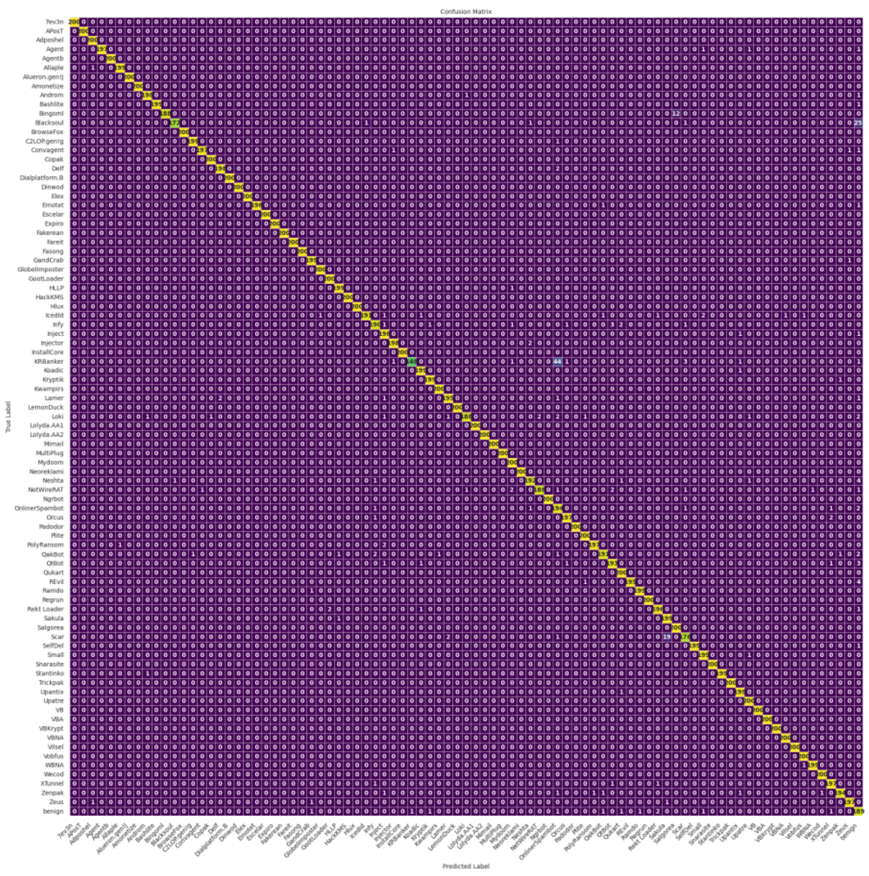


Fig. 11: CM of 86 families of malware after using hyperparameter tuning of the balanced dataset

A balanced dataset and minimizing the kernel and stride sizes make it possible to minimize the loss. However, computing time and cost will increase, and accuracy will become higher. When using balanced color images with fifteen epochs, the training and testing accuracy is better in the fine-tuned EPDL model shown in Fig. 10, but the result is not good in Fig. 12. It observes that the testing process accuracy and loss curves for balanced RGB color images were stable after less than seven or nine epochs. The confusion matrix (CM) is depicted in Fig. 9, where statistical evaluations visualize and summarize the performance of 86 families of malware.

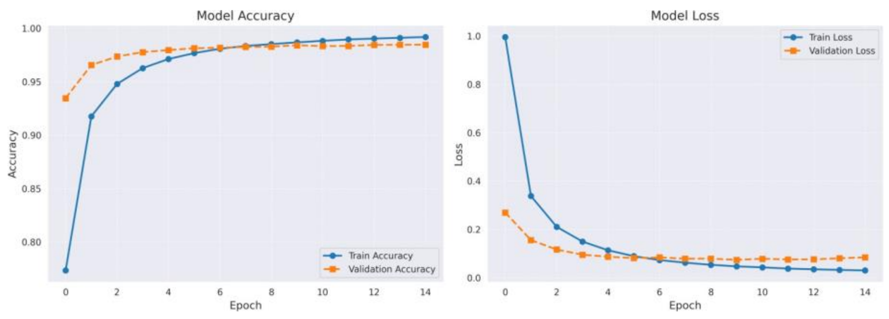


Fig. 12: The training and validation accuracy and loss of imbalanced image dataset

The following Fig. 13 shows the comparison among various models, such as EfficientNetB2, MobileNetV2, RegNet, ResNet50, and SwinTransformer, and statistical results, such as accuracy, precision, recall, and F1-score of proposed Parallel-CNN. It is shown that the proposed model precession is higher than other models.

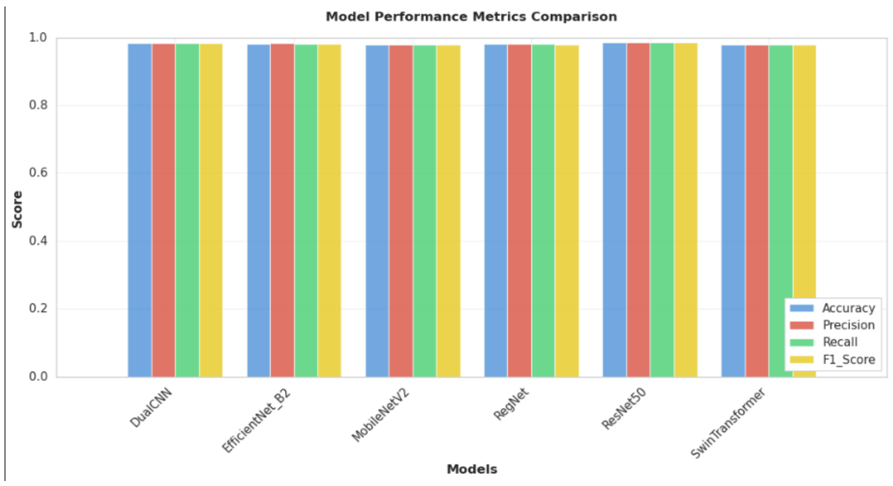


Fig. 13: Performance matrices comparison among various models

Fig. 13 illustrates that the accuracy, precision, recall, and F1-score vary among the comparison models of Efficie ntNetB2, MobileNetV2, RegNet, ResNet50, SwinTransformer, and the proposed Parallel-CNN. Inference time is the time for an EPDL model to interpret fresh input and generate an estimation quickly compared with other models. The Fig. 14 shows the proposed malware dataset is tested with the following CyberNet, BayesianCNN, EfficientNet_B2, MobileNetV2, RegNet-Unbalanced, ResNet50, and SwinTransformer models, including the proposed model, whose accuracy varies from 99% to 97%, which may validate the dataset.

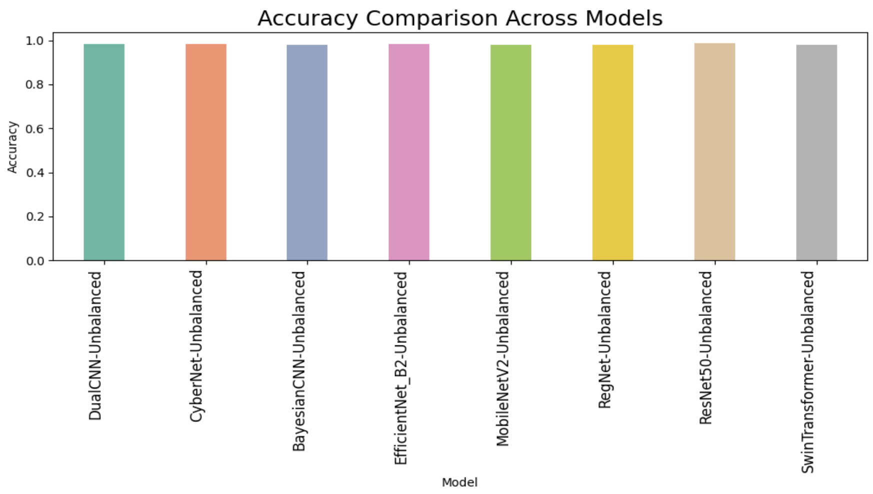


Fig. 14: Accuracy comparison of created new malware dataset with 7 models, including proposed parallel CNN

The following Table 6 shows the cross-validation accuracy from fold 0 to fold 9, where the average accuracy is 97.96%.

Table 6: K-Fold Validation accuracy results

Sl	**Fold No.(k)**	**Accuracy**
1	Fold 0	97.83%
2	Fold 1	98.16%
3	Fold 2	97.74%
4	Fold 3	97.74%
5	Fold 4	98.04%

6	Fold 5	97.90%
7	Fold 6	97.99%
8	Fold 7	97.83%
9	Fold 8	97.94%
10	Fold 9	97.99%
	Average	97.96%

Table 7 displays the results of CNN and EPDL performance comparisons. It attains superior performance when using the EPDL profiles to construct a classifier. This implies that EPDL can explore a sequence and is required to develop a more sophisticated classifier with a more robust identification of malware features utilizing the separate dataset. To evaluate the performance of the proposed model, which was created utilizing a parallel CNN, the 10-fold cross-validation and a standalone testing dataset were used. The approach yielded better results and significantly improved all the common assessment variables compared to other cutting-edge neural networks, illustrated in Table 7.

Table 7: Comparative performance between the proposed classification method and ANN

Method	**Cross-validation (Average)**				**Independent (Average)**			
	accuracy	precision	recall	f1-score	accuracy	precision	recall	f1-score
EPDL	97.96	97.75	98.74	97.74	99.00	99.00	99.00	99.00
CNN	96.13	96.00	96.24	96.18	97.21	97.30	98.00	97.40

The following malware classification experiment utilizes the end-to-end EPDL deep learning technique. It begins by extracting a 3-channel RGB representation from the raw pixel data associated with each malware case, corresponding to pairs of bytecode and metadata files [8]. The dataset generated from these features includes the target class labels for each malware family in the examples. Next, these features and labels are employed for 10-fold cross-validation. This process involves dividing the data vectors into ten (10) folds and removing 10% of the vectors to form each training set labeled as T-1, T-2, T-3, T-4, T-5, T-6, T-7, T-8, T-9, and T-10. Subsequently, the optimization algorithm, as described in batch normalization, where hyperparameters are fine-tuned, enhances the Harmony Search (EHS) algorithm. The optimization, an improved version of gradient descent learning, adjusts the parameters and optimizes HS for the end-to-end EPDL model for each fold. After training, each fold is evaluated, and the accuracy results are recorded. Finally, the average and top accuracy results are compared with previous research findings. The methodology enables the precise identification and exploitation of new malware datasets for malware identification. The graphical representation is shown in Fig. 15. Folds 0, 5, 7, and 8 give slightly lower results than Folds 1, 2, 3, 4, 6, and 9. The following Fig. 15 shows the comparison of the 10-fold cross-validation average accuracy, precision, recall, and F1-score of the proposed dataset and the EPDL model.

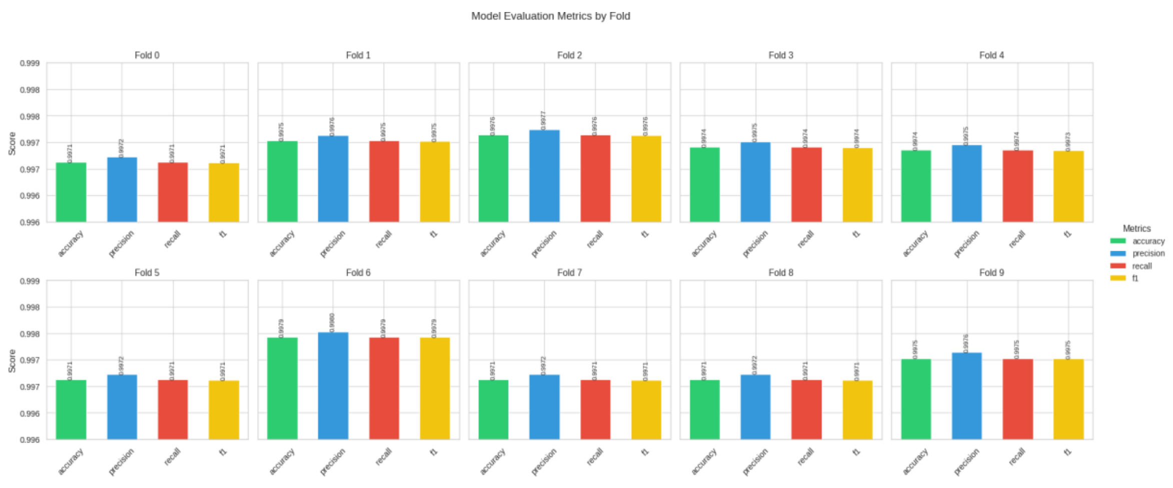


Fig. 15: Comparison each of 10-Fold cross-validation accuracy, precision, recall and F1-score

The experiment in this section demonstrates the EPDL-MD algorithm's superiority in CNN superparameter tuning.

When compared to seven other models, including CyberNet, BayesianCNN, EfficientNet_B2, MobileNetV2, RegNet-Unbalanced, ResNet50, and SwinTransformer models, as well as other cutting-edge CNNs, the EPDL-MD technique can produce superior results than the same type. Since the EPDL-MD algorithm optimizes the specified basic network acceptable parameters, the technique's enhancement impact is only noticeable if the initial network is appropriate for the tuning challenge. EPDL was selected as the fundamental structure of networks for this study for two distinct reasons. In the first place, it is a conventional deep-learning parallel CNN and a good illustration of how to optimize superparameters using this approach. Second, it has a more straightforward structure and less computation cost under a similar impact. Nevertheless, another area for improvement is selecting the right parallel CNN model hyperparameters to complete the related jobs. Consequently, this study contributes to the hyperparameter optimization issue for a particular parallel CNN.

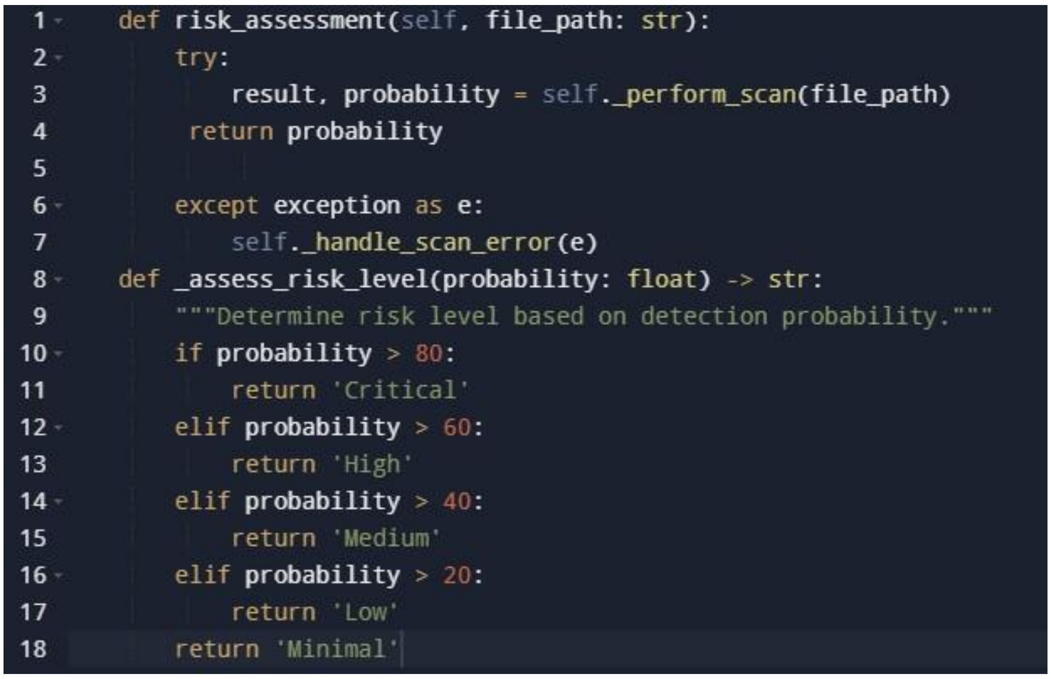
5.0 Response Software Threat Level Classification

Critical Risk (> 80% confidence) - This level signifies an immediate and severe threat that poses a substantial risk to system integrity and requires urgent intervention. Action must be taken without delay to mitigate the potential impact.
High Risk (61-80% confidence) - This threshold has significant potential for system compromise. While not as urgent as critical risks, these threats demand immediate attention to prevent escalation.

Medium Risk (41-60% confidence) - This classification indicates the detection of potential vulnerabilities. Although these threats may not require immediate action, they should be closely monitored and evaluated for possible remediation.

Low Risk (21-40% confidence) - This poses a minor threat with limited potential impact on the user or device. Routine monitoring is advised, but immediate action is not required.

Minimal Risk (≤ 20% confidence) - It indicates a negligible threat level. Routine surveillance is advised in such cases, as the threat does not warrant urgent response measures. The risk assessment algorithm is shown in the following Fig. 16.



```python
def risk_assessment(self, file_path: str):
    try:
        result, probability = self._perform_scan(file_path)
     return probability

    except exception as e:
        self._handle_scan_error(e)
def _assess_risk_level(probability: float) -> str:
    """Determine risk level based on detection probability."""
    if probability > 80:
        return 'Critical'
    elif probability > 60:
        return 'High'
    elif probability > 40:
        return 'Medium'
    elif probability > 20:
        return 'Low'
    return 'Minimal'
```

Fig.16: Risk assessment level in Python based on NIST 800-30 Guideline.

Binary malware files or code can be transformed into octal, hexadecimal, or decimal data to create 3-channel RGB images in a two-dimensional pixel matrix. Each of the three color channels (Red, Green, and Blue) contains values ranging from 0 to 255. Combining these three colors in various ways makes representing one of 16,777,216 potential colors of pixel features possible. These picture pixels generate the feature map, and depending upon the feature map classifier, it decides against a malware file. These byte sequences or instruction sequences are pixel values (1, 75,223,150,101), which make the feature map, and depending upon the feature map, the model responds in deep learning probability, and these probabilities are used to calculate the level of risk of each malware

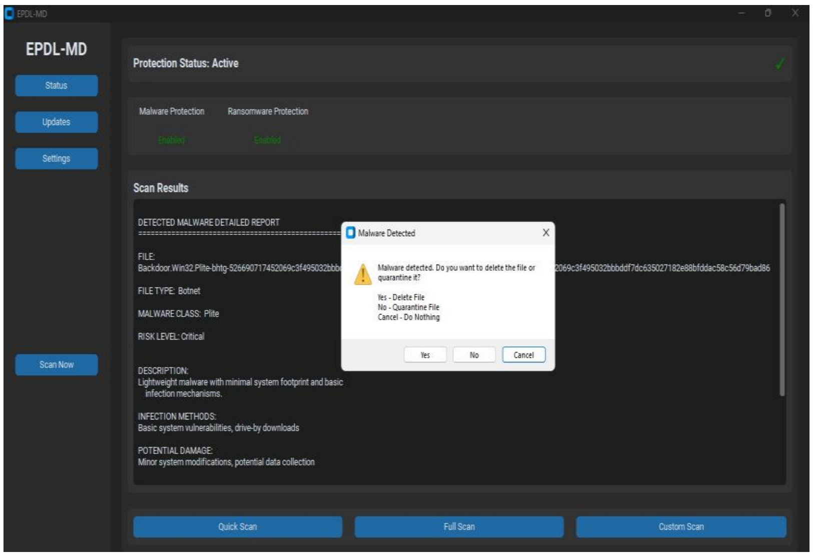
5.1 Graphical User Interface (GUI) of Response Software

The Graphical User Interface (GUI) is meticulously designed to enhance user interaction and experience through a comprehensive and intuitive structural Python 3.11 with TK framework [30]. The interface prioritizes user engagement by implementing a streamlined sidebar navigation system that facilitates seamless interaction with the malware detection application.

The primary interface is strategically organized into distinct functional zones. The main content area provides a comprehensive workspace for detailed scanning and analysis operations. Dynamic results display mechanisms enable real-time information presentation, while integrated status indicators offer immediate visual feedback on system processes and detection outcomes. Interactive capabilities are central to the interface design. Users can select from multiple scanning modes, including Quick, Full, and Custom configurations, providing flexibility in malware detection approaches. A configurable settings panel allows for personalized system optimization, enabling users to adapt the detection parameters to specific requirements. Immediate feedback mechanisms ensure users remain informed throughout the scanning and analysis process.

Result visualization is a critical component of the interface's design philosophy. The system incorporates progressive scan tracking, allowing users to monitor real-time detection progress. Detailed malware intelligence is presented through comprehensive visual representations, including risk assessment graphics that communicate potential threats effectively. The interface generates actionable mitigation recommendations, empowering users to make informed decisions about detected threats.

In Fig. 17, when malware is detected, the scan results show a detailed report with the file name of the malware family, the type of malware Botnet is in this case, and its risk level, which is also critical. It was used to take the footprint of the system. This report also shows the infection methods, such as drive download or using basic system vulnerabilities. The potential damage shows that the malware can minorly modify the system and may potentially collect the system's data.



Protection Status: Active ✓

Malware Protection: Enabled
Ransomware Protection: Enabled

Scan Results

```
DETECTED MALWARE DETAILED REPORT
==================================================

FILE:
Backdoor.Win32.Plite-bhtg-526690717452069c3f495032bbb…2069c3f495032bbbddf7dc635027182e88bfddac58c56d79bad86

FILE TYPE: Botnet

MALWARE CLASS: Plite

RISK LEVEL: Critical

DESCRIPTION:
Lightweight malware with minimal system footprint and basic
  infection mechanisms.

INFECTION METHODS:
Basic system vulnerabilities, drive-by downloads

POTENTIAL DAMAGE:
Minor system modifications, potential data collection
```

Malware Detected

Malware detected. Do you want to delete the file or quarantine it?

Yes - Delete File
No - Quarantine File
Cancel - Do Nothing

Fig. 17. User Notification Results on Scanning and Response Ask from user

The malware detection and response system is a sophisticated cyber security solution that integrates advanced deep learning methodologies with an intuitive user interface. By combining intelligent threat detection algorithms, comprehensive reporting capabilities, and optimized performance characteristics, the system provides robust protection mechanisms suitable for diverse computing environments [31].

6 Conclusion

In this paper, the researchers highlight that malware detection is a critical issue due to obfuscation techniques used by the creator of malware. If an attacker finds any vulnerability in a device's operating system (OS), they can gain control of the system quickly, so the proposed software can also detect the vulnerability of the OS. The proposed model classifier takes a long time for a large number of image datasets to train but achieves high accuracy; it could be more effective than traditional antivirus solutions. However, using a small dataset might result in the loss of many malware features, leaving the model vulnerable though trained fast. Conversely, employing a lightweight CNN model with fewer layers can enhance the model's performance within a specific dataset, but it may be less effective in detecting obfuscation types of malware. Previous research has shown that high-speed performance with limited layers and pre-built image-based datasets can yield high accuracy, but it fails to protect a system in real-time. However, malware developers are rapidly evolving their techniques and utilizing AI for attacks. Therefore, the researchers aim to detect and respond to existing malware with high precision, ensuring the system's security and the models' reliability by pattern-matching the pixels of malware. Though the code is different from the attacker's, the logic is similar, and the same features are used in the image to detect malware. If it becomes feasible to train the model with all available novel malware image datasets, it could also predict unknown malware or zero-day attacks, though here, we tried to collect 86 malware families. Subsequently, the parameters of DL were optimized by EHS to develop an EPDL model for malware detection, which focuses on image-based malware classification and aims for a high accuracy of 99%. The model created by the researchers benchmarked against the accuracy rates of previous researchers through a confusion matrix and cross-validation. Finally, anti-malware response software was developed using the Python Tkinter framework. This paper contribution culminated in the creation of more effective anti-malware software to address the future from mobile to large CPS. Future researchers plan to develop a model that works for any operating system to detect and prevent any possible malware universally.

This paper addresses mainly four limitations related explicitly to deep learning for malware detection:

(1) Image-based malware creation and detection require technical expertise to perform feature selection/extraction/engineering on digital image processing, costly training, and testing to adapt to changing threat landscapes, supervised learning requiring labeled data, over-fitting and under-fitting, and identification obfuscation processes (e.g., mimicking benign behaviors).
(2) Insufficient image-based malware training data exists in the malware detection area, and the network has millions of parameters; the researcher needs a lot of data to get an optimal set of parameters, and cost and time are major factors. The vast computing power required—even if managed from various sources, a lot of data—and training generally require multiple iterations and take a toll on computing resources.
(3) Limited learning efficiency due to the high false alarm rate identified in the existing deep learning models, which require too much time, power, and machines in cluster mode.
(4) There is not enough up-to-date malware image data set; the researcher would need to generate a new image dataset because the malware creator has rapidly changed its attacking code policy. There is a lack of recognized deep learning models for malware detection that can be used as a transfer learning model to train the latest datasets to validate, so it's challenging to build a model and tune the hyperparameter several times parallel with a novel malware dataset.

Acknowledgments The authors would like to express their gratitude to the Ministry of Higher Education (MOHE) Malaysia and Universiti Sains Islam Malaysia (USIM) for the support and facilities provided. This research paper project is funded by FRGS/1/2024/ICT07/USIM/01/1).

REFERENCES

[1] A. Marengo, "The Future of AI in IoT: Emerging Trends in Intelligent Data Analysis and Privacy Protection," *MDPI AG*, Dec. 2023, doi: 10.20944/preprints202312.2184.v1.

[2] M. A. Uddin, S. H. Sonali, M. S. Rahman, and M. S. Ahsan, "Deep Learning for Agile Malware Detection," *2024 IEEE Region 10 Symposium (TENSYMP)*, pp. 1–6, Sep. 2024.

[3] J. Zhang, X. Yu, X. Lei, and C. Wu, "A novel deep LeNet-5 convolutional neural network model for image recognition," *Computer Science and Information Systems*, vol. 19, no. 3, pp. 1463–1480, 2022.

[4] M. GUVEN, "Leveraging deep learning and image conversion of executable files for effective malware detection: A static malware analysis approach," *AIMS Mathematics*, vol. 9, no. 6, pp. 15223–15245, 2024.

[5] M. D. Shelar and S. S. Rao, “Enhanced capsule network-based executable files malware detection and classification—deep learning approach,” *Concurrency and Computation: Practice and Experience*, vol. 36, no. 4, Oct. 2023.

[6] Zhang, X. Yu, X. Lei, and C. Wu, “A novel deep LeNet-5 convolutional neural network model for image recognition,” *Computer Science and Information Systems*, vol. 19, no. 3, pp. 1463–1480, 2022.

[7] S. Parvez, “Deep Learning Model for Image Classification Using Convolutional Neural Network,” *International Journal of Science and Research (IJSR)*, vol. 11, no. 8, pp. 132–137, Aug. 2022.

[8] Q. Liu and Z. Liu, “Intelligent recognition of subgrade damages combining GPR signals and CNN models,” *International Conference on Image, Signal Processing, and Pattern Recognition* (ISPP 2024), p. 62, Jun. 2022.

[9] T. Muralidharan, A. Cohen, N. Gerson, and N. Nissim, “File Packing from the Malware Perspective: Techniques, Analysis Approaches, and Directions for Enhancements*,” ACM Computing Surveys*, vol. 55, no. 5, pp. 1–45, Dec. 2022

[10] C. S. Islam and S. H. Mollah, “The X-Layer Optimization in CRN Using Deep Q-Network for Secure High Speed Communication,” 2019 11th International Conference on Information Technology and Electrical Engineering (ICITEE), pp. 1–6, Oct. 2019

[11] R. Vinayakumar, K. P. Soman, and P. Poornachandran, “Evaluation of Recurrent Neural Network and its Variants for Intrusion Detection System (IDS),” Deep Learning and Neural Networks, pp. 295–316, 2020

[12] J. Song, J. Xiao, C. Tian, Y. Hu, L. You, and S. Zhang, “A Dual CNN for Image Super-Resolution,” *Electronics*, vol. 11, no. 5, p. 757, Mar. 2022.

[13] Li, J. Mi, W. Li, J. Wang, and M. Cheng, “CNN-Based Malware Variants Detection Method for Internet of Things,” *IEEE Internet of Things Journal*, vol. 8, no. 23, pp. 16946–16962, Dec. 2021

[14] B. Sun, C. Wu, and M. Yu, “Spectral Reconstruction for Internet of Things Based on Parallel Fusion of CNN and Transformer,” *IEEE Internet of Things Journal*, pp. 1–1, 2024

[15] B. Esmaeili, A. Azmoodeh, A. Dehghantanha, G. Srivastava, H. Karimipour, and J. C.-W. Lin, “A GNN-Based Adversarial Internet of Things Malware Detection Framework for Critical Infrastructure: Studying Gafgyt, Mirai, and Tsunami Campaigns,” *IEEE Internet of Things Journal*, vol. 11, no. 16, pp. 26826–26836, Aug. 2024

[16] Y. F. A. Gaus, N. Bhowmik, S. Akcay, P. M. Guillen-Garcia, J. W. Barker, and T. P. Breckon, “Evaluation of a Dual Convolutional Neural Network Architecture for Object-wise Anomaly Detection in Cluttered X-ray Security Imagery,” *2019 International Joint Conference on Neural Networks (IJCNN*), pp. 1–8, Jul. 2019

[17] R. Alaa, H. Al-Libawy, and E. A. Hussein, “Low-Cost Blind Spot Detection System Based on Lite Object Detection Algorithm and Limited Resources Hardware,” *2024 4th International Conference of Science and Information Technology in Smart Administration (ICSINTESA)*, pp. 469–474, Jul. 2024

[18] C. S. Islam, M. Mohd Saudi, N. H. Zakaria, and M. Setiyo, “Malware Detection using Deep Learning (DL),” *Journal of Advanced Research in Applied Sciences and Engineering Technology*, pp. 253–273, Oct. 2024

[19] “1D-Convolutional Neural Network Architecture for Generalized Time-Segmentation Tasks,” Jan. 2023

[20] Y. Zhou, X. Liu, Z. Guo, Y. Zhou, C. Zhang, and J. Qian, “Deep learning or classical machine learning? An empirical study on line-level software defect prediction*,” Journal of Software: Evolution and Process*, vol. 36, no. 10, Jun. 2024

[21] C. S. Islam and Md. S. H. Mollah, “A Novel Idea of Malaria Identification using Convolutional Neural Networks (CNN),” *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, pp. 7–12, Dec. 2018

[22] G. Wang, J. Liu, J. Xie, Z. Zhu, and F. Zhou, “Joint multimodal entity-relation extraction based on temporal enhancement and similarity-gated attention,” *Knowledge-Based Systems*, vol. 304, p. 112504, Nov. 2024

[23] Q. Chen, J. Li, and X. Fang, “Dual triple attention guided CNN-VMamba for medical image segmentation,” *Multimedia Systems*, vol. 30, no. 5, Sep. 2024

[24] V. Malik, A. Khanna, N. Sharma, and S. nalluri, “Trends in Ransomware Attacks: Analysis and Future Predictions,” *International Journal of Global Innovations and Solutions (IJGIS),* Aug. 2024

[25] N. K. Mishra, P. Singh, A. Gupta, and S. D. Joshi, “PP-CNN: probabilistic pooling CNN for enhanced image classification,” *Neural Computing and Applications*, Dec. 2024

[26] R. E., “Design of Efficient Low Pass Infinite Impulse Response Filter Using Dynamic Regional Harmony Search Algorithm with Opposition and Local Learning and Comparison with Harmony Search Algorithm,” *Revista Gestão Inovação e Tecnologias*, vol. 11, no. 4, pp. 1289–1302, Jul. 2021

[27] A. Shabani, X. Zhang, X. Chu, and H. Zheng, “Automatic Calibration for CE-QUAL-W2 Model Using Improved Global-Best Harmony Search Algorithm,” *Water*, vol. 13, no. 16, p. 2308, Aug. 2021

[28] P. Mao and K. Li, “OpenTOS: Open-source System for Transfer Learning Bayesian Optimization,” *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 5254–5259, Oct. 2024

[29] L. L. Thornton, D. E. Carlson, and M. R. Wiesner, “Predicting emerging chemical content in consumer products using machine learning,” *Science of The Total Environment*, vol. 834, p. 154849, Aug. 2022

[30] R. L. K. -, “Active Learning Framework for Python Programming,” *International Journal For Multidisciplinary Research*, vol. 6, no. 2, Apr. 2024

[31] J. Borrow, “Making Research Data Flow with Python,” Python in Science Conference, 2024. *NumFOCUS - Insight Software Consortium (ITK)*, Jul. 10, 2024