International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V72I10P128

Volume 72 Issue 10, 306-315, October 2024
© 2024 Seventh Sense Research Group®

Original Article

Deep Learning Approach for IoT Malware Detection and Classification based on Squirrel Search Algorithm and Convolutional Neural Network (IMD_SSACNN)

V. S. Jeyalakshmi[1], Krishnan Nallaperumal[2]

[1,2]*Centre for Information Technology and Engineering, Manonmaniam Sundaranar University, Abishekapatti, Tirunelveli, Tamilnadu, India.*

[1]*Corresponding Author : vsjeyalakshmiap@gmail.com*

Received: 13 June 2024 Revised: 05 October 2024 Accepted: 14 October 2024 Published: 25 October 2024

Abstract *- The number of Internet of Things (IoT) devices that are exposed to the public has been rising as more of these devices are connecting to the internet using their default settings. Due to the variety of designs and the IoT's limited computation and storage capacities, it is challenging to implement sufficient security measures, which makes it more susceptible to infection. Accurate IoT malware identification and family attribution are crucial in order to begin implementing attack mitigation/prevention tactics, which is why they are so important in order to reduce the threat. To prevent the risks caused by malicious code, various research has been done on the identification of IoT malware. It might be challenging to recognize the novel variant IoT virus that is being created rapidly, even though existing models might successfully identify hazardous IoT code found through static analysis. This research introduced a novel IoT Malware Detection with a Squirrel Search Algorithm and Convolutional Neural Network (IMD-SSACNN). IoT malware datasets are used to conduct an exhaustive analysis of the suggested technique. IMD-SSACNN is able to lessen the damage that malware infestation brings to IoT devices by examining and analyzing the massive volume of behavior data generated by dynamic analysis. The experimental findings show that the suggested IMD-SSACNN is the preferred approach since it has a greater detection rate than the earlier malware detection algorithms.*

Keywords *- Deep Learning, Malware detection, IoT malware, Mitigation, Convolution Neural Network.*

1. Introduction

The IoT has been widely used to improve a variety of areas of the human living environment, including healthcare, transport, etc. IoT has been incorporated into many facets of our daily lives. Simultaneously, there is growing concern about the security of IoT applications [1]. 50 billion IoT devices are expected to be in operation globally by 2030, according to predictions [2]. Despite their advantages, the growing prevalence of malware designed specifically for the IoT that aims to use compromised IoT devices (such as weak authentication) to coordinate massive cyberattacks has posed a serious threat to the broader Internet ecosystem. The application of traditional security procedures is, however, impractical with IoT due to their networked and autonomous nature, as well as their limited computation and storage capacities. This means that malware attacks can occur on a large number of IoT devices. IoT malware threats have been rapidly increasing. IoT device attacks are typically not sophisticated, but they are sneaky, so users are not aware that their gadgets are being abused. More than 120,000 different malware variants of IoT malware seen in 2017, targeted IoT devices in the first half of 2018, according to the Kaspersky Lab IoT report [3]. A risky trend is continuing with the exponential growth of malware targeting IoT devices. Less than 2% of new malware varieties, however, differ in their coding from known malware [4]. It shows that the majority of malware variants from the same family share functionality to some extent, which is a crucial foundation for malware classification. Accurately identifying and categorizing the rapidly multiplying IoT malware variants is crucial. Research has been done on virus detection by feature learning and classification in an attempt to reduce the harm caused by malware infection by shielding IoT devices against new and unique malware attacks [5, 6]. The process of detecting malware is usually divided into two phases by studies: analysis and detection [7]. Malware characteristics were obtained during the analysis stage. The detection step is conducted to search for malware in the content that was inspected after the malware analysis. Malware can be prevented from infecting further IoT devices by recognizing known, unknown, and variant IoT malware using malware analysis and detection tools. In this research, a novel deep learning framework, IoT Malware Detection and Classification based on Squirrel Search Algorithm and Convolutional Neural Network (IMD-SSACNN), is proposed. The framework integrates the SSA, a nature-inspired optimization method, with a Convolutional Neural Network (CNN) to improve malware detection and classification efficiency. The SSA is used to optimize hyperparameters, ensuring that the CNN model achieves high accuracy and minimal false positives in detecting and categorizing IoT malware. By leveraging the SSA's ability to explore and exploit the search space dynamically, the proposed model can effectively adapt to the diverse and evolving nature of malware in IoT systems. The CNN, known for its strong feature extraction capabilities, enhances the classification performance, making the model robust in distinguishing between benign and malicious IoT traffic.

This article suggests a dynamic IoT malware detection system and Classification based on a Squirrel Search Algorithm and Convolutional Neural Network (IMD-SSACNN). To simplify malware classification models and satisfies the security and defense requirements of various platforms. The suggested dynamic analysis in order to identify the IoT malware through a continuous analysis of malware behavior. Dynamic analysis tools are used to obtain the disassembled binary files in a virtual environment. Later, it was converted into image files for malware analysis. The remainder of the piece is organized as follows: In Section 2, the research approaches for assessing and recognizing IoT malware are examined. Section 3 describes the IMD-SSACNN presented in the paper. In Section 4, the suggested model's performance evaluation is discussed. The paper's findings and recommendations for additional research are provided in Section 5.

2. Related Works

This section is a survey of the literature on IoT malware detection. Given a summary of the current DL-based approaches for IoT malware classification. Vasan et al. [8] have suggested a novel architecture based on ensemble Convolutional Neural Networks (CNNs) that are capable of effectively detecting malware that is both packed and unpacked. A set of CNN architectures enables the extraction of features with higher quality than conventional techniques. According to experimental findings, IMCEC is especially suited for malware detection. It might be able to achieve low false alarm rates and high detection accuracy using malware raw-input. The outcome shows over 99% accuracy for malware that has been unpacked and over 98% accuracy for malware that has been packed. A novel malware-detection model employing opcode sequences and a convolutional recurrent neural network is presented by Jeon et al. [9]. An executable file is seen statistically as a collection of sequential machine codes. First, it has been covered how to use opcode sequences to look for malware theoretically. The Area Under the Curve (AUC) of the suggested model was 0.99, its True Positive Rate (TPR) was 95%, and its malware-detection accuracy was 96%. Kuang et al. [10] suggest a unique deep neural network-based approach for detecting visual malware. First, samples of executable files are gathered and disassembled using technology to create bytes and ASM files. After that, the samples are further transformed into three-channel RGB images by employing data augmentation and visualization methods to extract high-dimensional intrinsic features from the data samples. Giacomo Iadarola et al. [11] describe a technique that uses an explainable deep learning model created by the authors to detect malware on Android and identify families, which relies on the representation of the application in terms of the photographs used as input.

Furthermore, this study demonstrates how the analyst might take explainability into account when evaluating various models. Averaging accuracy between 0.96 and 0.97, this model examined 8446 Android samples from six separate malware families and one additional family of trusted samples. Transformers on raw binaries for real-time malware categorization were studied by Lu et al. [12]. Two models that interpret the binary in two distinct ways are put forth in order to do malware categorization. In order to handle the raw data as a byte sequence, they suggested by using SeqConvAttn. Observed the latency for classification increases quadratically with the length of the input sequence, subsequently introduced ImgConvAttn. This low-latency substitute model employs Vision Transformer for malware classification using preprocessed images.

The two combined models into a two-stage framework that takes the file size into account in order to effectively utilize both models. The MalFSM framework is proposed by Kong et al. [13]. The machine learning classification is effectively applied and achieves a good accuracy rate after reducing the 735 opcode features in the Kaggle dataset to 16 using the feature selection approach. Then, fuse on metadata features (the number of file lines and file size) for a total of 18 features and evaluate the chosen features to examine the relationship between the malicious samples' opcode features. The extensive tests applied to the Microsoft Kaggle malware dataset demonstrates that the maximum classification accuracy of MalFSM achieved 98.6% and the classification time is only 7.76 s.

Table 1. An overview of earlier research on anomaly detection

S.No.	Reference	Model	Accuracy
1	[8]	ECNN	99%
2	[9]	Convolutional Recurrent Neural Network (CRNN)	96%
3	[10]	SERLA	98%
4	[11]	Grad-CAM	97%
5	[12]	ImgConvAttn	93.42%
6	[13]	MalFSM	98.6%

3. Proposed Methodology

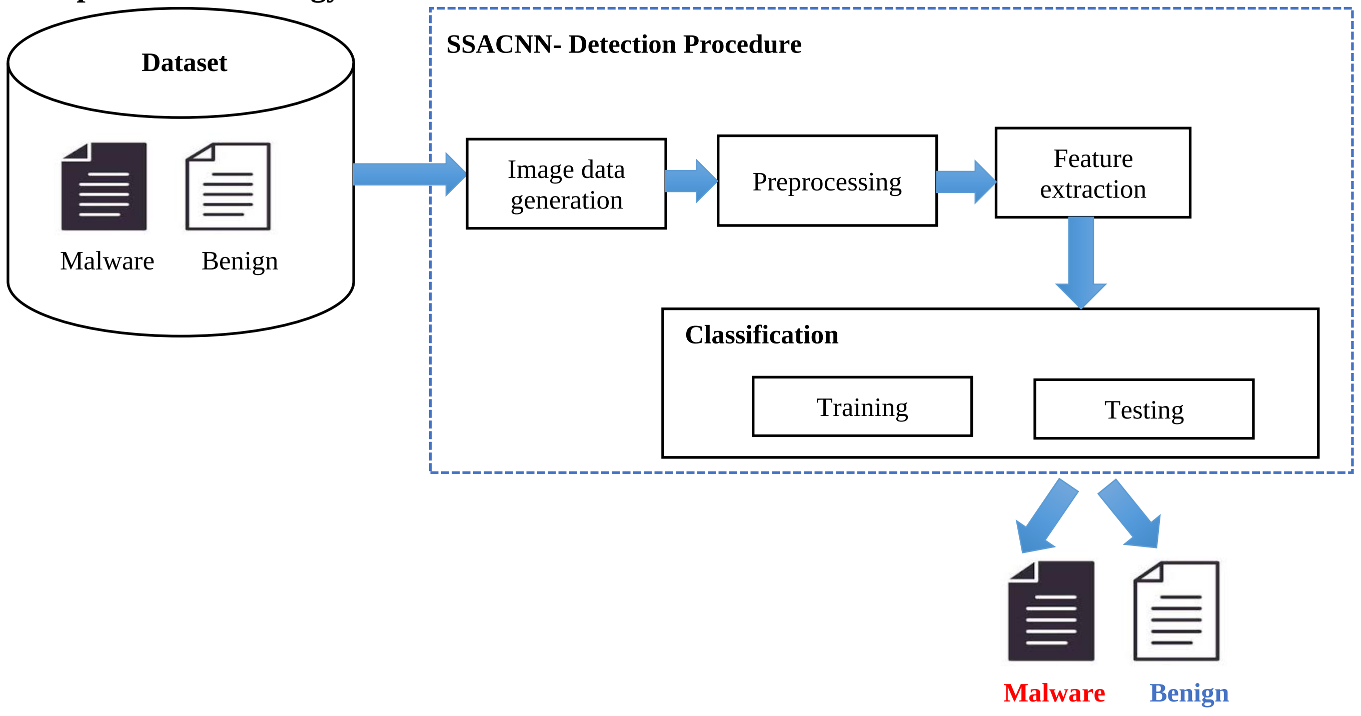

Fig. 1 Proposed SSACNN based malware detection architecture

This study addresses the challenge of IoT malware classification through the analysis of IoT malware binaries. Because IoT devices lack secure protocols and security standards, they are more vulnerable to malware assaults. As a result, protecting data in Internet of Things networks is a difficult but necessary undertaking. IoT-malware byte sequences are examined in the paper. For the experiment, byte sequence is taken into consideration as a characteristic because it can be built into a detection model with little computational complexity and is platform-independent. In this research, the IoT malware is classified using the L based approach that can identify and classify IoT malware families in order to address the limitations mentioned above. The goal is to create and evaluate a classification technique that increases overall classification accuracy by combining many modalities and proactively discovering characteristics from different malware binary representations. The proposed malware classification model is described in the section. The section goes into great detail on the dataset used for the research, data preparation, the suggested SSACNN approach, and experimental design. The working steps of the proposed SSACNN architecture on malware detection are illustrated in Figure 1.

3.1. Dataset

The BIG15 dataset is utilized for experimentation and assessment of the suggested technique. Microsoft makes this dataset available on the Kaggle platform for competition purposes. Microsoft offers this for competition purposes. There are two sets of it, each with 10868 training samples and 10873 test samples, totaling 21,741 different malware kinds [14]. This dataset takes up approximately half a terabyte when it is uncompressed. In this experiment, just the train data are used; the test sample lacks labels. A class, an integer that designates one of nine potential malware family names, and an id, a hash value of 20 characters that serves as the file's unique identity, are both present in every malware file. The number of instances for each malware class of the BIG 15 dataset is illustrated in Table 2. Without the PE header, each sample of malware is just a raw hexadecimal representation of the file's binary data. A metadata representation is additionally included, which contains information on function calls, embedded text, etc., that was collected using a disassembler tool. The work focuses on using deep learning approaches to classify malware based on the content of raw binary files.

Table 2. Number of instances in the dataset for each malware class

Class	Number of Instances
Ramnit	1541
Lollipop	2478
Kelihos_ver3	2942
Vundo	475
Simda	42
Tracur	751
Kelihos_ver1	398
Obfuscator.ACY	1228
Gatak	1013

3.2. Data Pre-processing and Feature Extraction

Deep learning has the benefit over other machine learning approaches in that it may be used on raw data without the need for explicit feature engineering in a particular area. The objective is to effectively categorize malware without requiring specialized knowledge or time-consuming procedures to find and extract malware signatures. This serves as a major motivator for the work. For the deep learning strategy to parallelize the computation for training and testing the models, each file must have a specified size. On the other hand, malware can exist in a variety of file sizes. In addition to being the same size, the deep learning approaches need that the size be restricted from a computational standpoint in order to keep the model training procedure feasible with standard hardware. To preserve as much of the original structure as possible because the deep learning algorithms are built to identify and detect similar patterns and structures in malware file data, even while padding and truncation have been employed to standardize the file size.

3.3. Image Data Generation from Byte Data

Before the resizing and normalization, byte files are converted into image data. Malware is represented as binary or assembly mnemonics at the machine level. Byte files [15], which are effectively compiled binaries, serve as malware representations. With the aid of programmers like Interactive Disassembler, the compiled binary must first be transformed into equivalent mnemonic sequences as the first stage in the categorization process (IDA). Then, mnemonics are transformed into hexadecimal digit combinations and combined, as illustrated in Figure 2. Then, set the image's preset width and height to 1024x1 to ensure that hexadecimal sequences are represented appropriately in pixels. Such an image had 2 hexadecimal digits per pixel, each of which represented a byte. Later, this image was scaled using conventional bicubic interpolation to match the SSACNN input channel size.

3.4. Resizing and Normalization

Due to the input size's flexibility, it can be adjusted to fit the algorithms employed in this study. For data normalization, the min-max scaling strategy is used in addition to regular scaling [16]. Normalization keeps the model from becoming stale and is essential for learning on neural networks. The following is a definition of a Min-Max Scalar (MMS).

$$MMS = \frac{DS - DS_{min}}{DS_{max} - DS_{min}} \quad (1)$$

DS signifies a data sample within a column, whereas DS_{min} and DS_{max} indicate the minimum and maximum data samples in that column. Post Min-Max normalization, the values reside inside the range [0, 1], which is sometimes a preferred attribute for input [16].

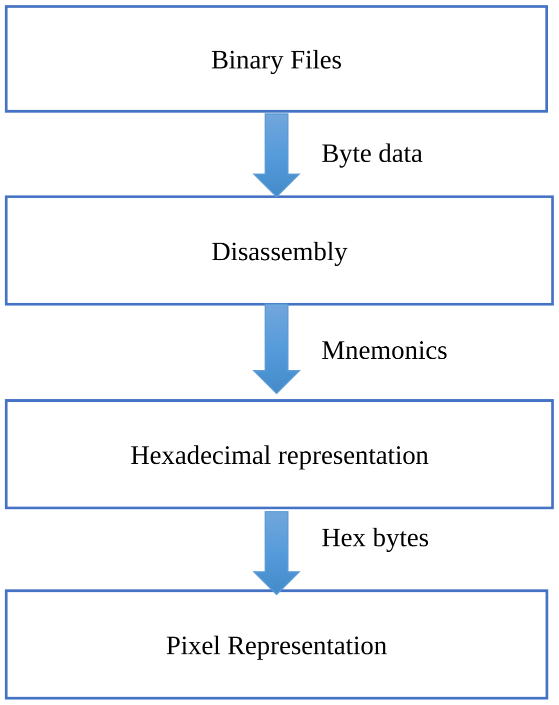

Fig. 2 Steps involved in byte into image conversion

The data surrounding a mean with a single standard deviation are standardized by traditional scaling, given by the following equation

$$cs = \frac{DS - \mu}{\sigma} \quad (2)$$

3.5. Autoencoder

An autoencoder is an artificial neural network designed to identify a data representation. Its principal application is in dimensionality reduction. They may acquire fundamental data representations and how to revert these representations to the original data.

An autoencoder has three fundamental components:

- The objective of encoding architecture is to reduce the dimensionality of incoming data. It often manifests as layers with a reduced number of nodes.
- Subordinate representation of data subsequent to its traversal through the encoding framework.
- A growing quantity of layers in a decoding architecture is employed to transform data from a low-level representation to its original form.

L1 and L2 weight regularization coefficients are incorporated into the loss function to enhance generalization in autoencoder training. Regularization is a fundamental machine learning principle. Its goal is to prevent overfitting of the model. It is utilized in the cost function minimization process. To minimize the costs of L1 regularization, sums of weights are added to them. As a result, it generates incredibly sparse matrices with plenty of zeros. In contrast, the cost function is combined with the weights' sum and square in the L2 regularization. The cost function additionally employs L2 regularization to prevent overfitting.

4. Feature Extraction

Convolutional layers of CNNs are trained with convolutional autoencoders to achieve feature extraction. The original byte data are utilized to train two Convolutional Autoencoders (CAE). The original encoded data from the first encoder is input into the second autoencoder. For fine-tuning, two completely linked layers of a CNN are employed as convolution layers, with the pre-trained encoding layers from CAE. The optimal model state with the lowest log loss for validation data is saved throughout fine-tuning. Following training, the best model is applied to the entire dataset (train, validation, and test) to produce nine probabilities for each malware sample as features.

4.1. Squirrel Search Algorithm

SSA is one of the more sophisticated Swarm Intelligence (SI) models developed using squirrel foraging behavior [17]. The SSO approach is developed by flying squirrel's foraging activity and a successful method for migration involved using small animals. A squirrel may be simulated in a multi-dimensional search area while looking for food in a population-based scheme with many squirrels. Different parameters are assigned to different places of squirrels in the SSA technique. Individual squirrels also shift their locations in order to find the best option. Some of the significant elements of SSA are population sizes P_S, maximum iteration value I_{mx}, likelihood of predator presence P_p, decision variables value d, gliding constants g, scaling factors F, and upper and lower limits to decision variables FLS_u and FLS_l. A random position for the squirrel is loaded from the search space.

$$FLS_{a,b} = FLS_L + r() * (FLS_u - FLS_L),$$
$$a = 1,2,\cdots,P_S \; and \; b = 1,2,\cdots,d \qquad (3)$$

Where rand () designates an arbitrary value in the range [0, 1], a squirrel position's fitness measure $fm = fm_1 fm_2, fm_{P-S}$ was processed by substituting FF for the decision variable: Next, the fitness measure of a squirrel position is used to assess the quality of food sources as follows:

$$[sorted_{fm}, sorted_{index}] = s(f) \qquad (4)$$

The array is sorted ascending after recording the fitness values of each flying squirrel's position. Food supplies were organized and included hickory trees, regular trees, and oak trees (acorn nuts). The hickory nut tree (FLS_{hn}) was thought to be the best food source (lowest fitness), followed by acorn nut trees (FLS_{an}), and the remaining trees are known as "regular trees" (FLS_{nt}). As previously mentioned, there are three potential consequences when flying squirrels engage in dynamic foraging. In all instances, it is believed that in the absence of predators, the flying squirrel glides and efficiently forages the forest for its chosen sustenance; nevertheless, when a predator is present, it becomes vigilant and must undertake a swift, aimless search for a concealment location. This mathematical model may represent dynamic foraging activity.

- Scenario 1-Flying squirrels may move from acorn nut trees to hickory nut trees (FlS_{an}). The new position of the squirrel may be determined as follows:
- Flying squirrels have the potential to migrate from acorn nut trees (FlS_{an}) to hickory nut trees. In this case, the following is how to find the new squirrel position:

$$FLS_{an}^{t+1} = \begin{cases} FLS_{an}^{t+1} + Rg_d \times g \times (FLS_{hn}^{t} - FLS_{an}^{t}) & N_1 \geq P_p \\ RL & otherwise \end{cases} \qquad (5a)$$

In this context, RL denotes the random location, FS_{hn} indicates the position of the flying squirrel that has arrived at the hickory nut tree, Rg_d represents the random gliding distance, and t signifies the current iteration. The glide constant g within the mathematical model facilitates the equilibrium between exploration and exploitation. The value significantly influences the performance of the proposed algorithm. The value of g is established at 1.9 in the present study, determined through thorough analysis.

- Scenario 2: To fulfil their daily energy needs, flying squirrels residing in regular trees (FS_{nt}) may migrate in proximity to acorn nut trees. The following are newly identified locations for squirrels:

$$FLS_{nt}^{t+1} = \begin{cases} FLS_{nt}^{t} + Rg_d \times g \times (FLS_{an}^{t} - FLS_{nt}^{t}) & N_2 \geq P_p \\ RL & otherwise \end{cases} \qquad (5b)$$

- Scenario 3: In situations where food availability decreases, certain squirrels that have been foraging on acorn nuts from prevalent tree species may transition to hickory nut trees, allowing them to store hickory nuts for future use. The following locations for squirrels have been identified:

$$FLS_{nt}^{t+1} = \begin{cases} FLS_{nt}^{t} + Rg_d \times g \times (FLS_{hn}^{t} - FLS_{nt}^{t}) & N_3 \geq P_p \\ RL & otherwise \end{cases} \qquad (5c)$$

$N_1, N_2 \; and \; N_3$ are the random values between [0, 1]. Flying squirrel foraging habits are influenced by the season, which changes often. The trapping is therefore eliminated in the local ideal outcome as a result of the implementation of the seasonal observation. The minimum value and seasonal constant Snc can be expressed as follows:

$$Snc_c^t = \sqrt{\sum_{k=1}^{d} \left(FLS_{an,k}^{t} - FLS_{hn,k}\right)^2}, t = 1,2,3 \qquad (6)$$

$$Sc_{mn} = \frac{10E-6}{365^{I/(I_{mn})/2.5}} \tag{7}$$

As the winter season increases $Snc_c^t < Sc_{mn}$, the squirrel loses its capacity for exploration, and the manner of looking for food sources and sites changes:

$$FLS_{nt}^{new} = FLS_L + L(d) \times (FLS_u - FLS_L) \tag{8}$$

The Lévy distribution is now used to transform the global search into an improved technique:

$$L(x) = 0.01 \times \frac{dr_i \times \alpha}{|dr_j|^{1/\delta}} \tag{9}$$

Where δ is a constant in the current work assumed to be 1.5, dr_i and dr_j are two randomly distributed values in the range [0, 1], and is calculated as:

$$\alpha = \left(\frac{\Gamma(1+\delta) \times sin\left(\frac{\pi\delta}{2}\right)}{\Gamma\left(\frac{1+\delta}{2}\right) \times \delta \times 2^{\left(\frac{\delta-1}{2}\right)}} \right)^{1/\delta} \tag{10}$$

Where $\Gamma(x) represents\ the\ value\ of\ (x-1)$

4.2. CNN

The CNN classifier was developed to lower the classification error rate caused by other feature extraction techniques that lacked the CNN method's level of robustness. This section describes the CNN and its intricate algorithm process. The layers of this classifier are to create a network of neurons depending on the dataset's relevant properties during the classifier's training phase. To extract the class label of the testing image frame, this computes the separation between the training set and the testing feature vector during the testing instance. The CNN is composed of three main layers: the fully connected layer, the max-pooling layer, and the convolutional layer.

During the CNN training phase, these layers were mostly engaged in the two main types of propagation, which are forward propagation and backward propagation. The feature's importance is evaluated using one of these two propagations, and the distance's compliance with the permissible range is ascertained using the other. That includes the mathematical model to express the network setup based on the input feature matrix and a hidden layer of NN. With the help of these five layers, CNN trains its features and predicts how relevant they will be.

This was considered for the dataset's training case for the input patterns of image frames. By determining the shortest path between the training feature set and the testing vector, the feature vector was subsequently verified from the intricate pattern during testing and was able to predict the class. The completely networked layer represents CNN's categorized output and is a part of the output stage. In order to create the small patches in CNN, the convoluted image frames have to be reassembled into a number of pre-defined blocks. The length of the features that need to be classified determines the size of the patch separation.

4.2.1. CNN Classifier Training Utilizing SSA

The utilization of the "back propagation" method exclusively for training a Neural Network (NN) may result in the solution being confined to a "local" optimum. The SSA optimization algorithm is utilized to identify the optimal solution by executing multiple cycles to address this issue. This article analyses the SSA method for achieving optimal weight initialization in CNN classifiers. The SSA search agents determine the weights of the "CNN" classifier through the use of actual numerical values.

The initial real integers are utilized to encode the three convolutional masks. The final actual value functions as the "seed measure" for the Random Number Generator (RNG), which is employed to initialize the CNN. Additionally, the 'CNN' classifier undergoes training exclusively through the back-propagation method for a minimum of Q1 epochs. Training the CNN classifier using the back-propagation technique significantly reduces the risk of data overfitting. The conventional feed-forward artificial neural network is also referred to as the back-propagation technique.

The back propagation method is employed to train the network, achieving the A1 epochs metric, which is considered the fitness value of the SSA search agent. This study investigates a specific set of weight initializations to identify the optimal weight initialization that results in an enhanced solution utilizing the SSA algorithm. To identify a solution that is more likely to represent the "global" optimal.

To utilize the gradient descent algorithm to enhance the search process and effectively identify the local optimum within a specified weight initialization basin. The advantage of employing the "gradient descent" method lies in its ability to decrease the time required to identify the "local" optimum from the "weight initializations" utilizing the SSA algorithm (basins). This process can reduce the number of candidates, known as "initial weights," that the SSA must evaluate to identify the "local" optimum. Algorithm 1 demonstrates the SSACNN working architecture.

Algorithm 1: SSACNN

```
Initialize the SSA population.
Calculate the SSA search agents' fitness metric and
use it to establish a "CNN" classifier.
Execution of SSA for a number of different cycles
using equations (1-4).
Calculate the final population of SSA.
Initialize CNN classifiers using the final SSA population.
Combine CNN classifier outputs (evidences) from SSA agents.
```

4.2.2. Experimental Results

The experiments designed to determine whether the hybrid framework for malware classification is effective are presented. In this section, the BIG15 dataset is used to compare the outcomes of the proposed model's performance analysis. Models are taught and assessed according to their intrinsic behavior and classification proficiency.

To preserve class distribution, stratified splitting was employed to partition the BIG15 dataset into a 75% training set and a 25% testing set, thereafter training the model. The metrics used for the evaluations are precision, recall, F-measure and Accuracy.

Precision (P)

Its definition is the ratio of real positive, relevant occurrences to all retrieved instances. It is supplied by:

$$P = \frac{TP}{\mathrm{TP + FP}} \tag{11}$$

Precision is a critical metric for evaluating the proportion of correctly identified positive instances relative to the total number of instances classified as positive. It is particularly important in scenarios where minimizing false positives is a priority. In this study, the CNN model achieved a precision of 98.39%, whereas the SSACNN model outperformed it with a precision of 99.18%. The improved precision of the SSACNN model demonstrates its effectiveness in reducing false positives, making it a more reliable classifier in this context.

Recall (R)

The sensitivity, as it is often known, is the proportion of accurately anticipated positive cases to the total number of positive events.

$$Recall = \frac{TP}{\mathrm{TP + FN}} \tag{12}$$

Recall, also known as sensitivity, measures the ability of a model to correctly identify all relevant instances (true positives) within the dataset. The recall for the CNN model was observed to be 98.57%, while the SSACNN model demonstrated superior performance with a recall of 99.28%. This higher recall indicates that the SSACNN model is more capable of detecting true positives, making it particularly effective in identifying positive instances without missing significant portions of the relevant data.

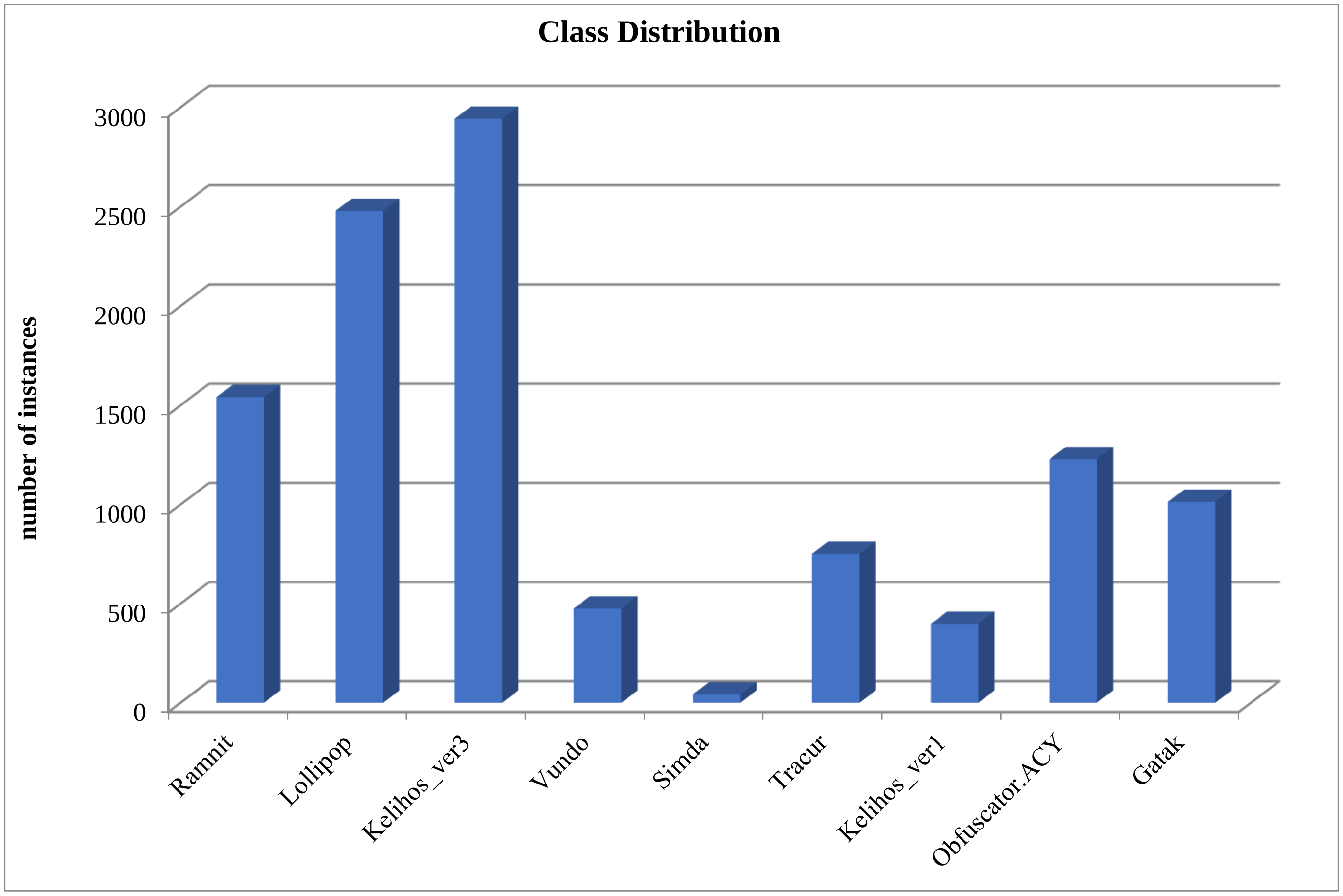

Fig. 3 Class distribution of microsoft malware dataset

Accuracy

It is the ratio of the number of correctly predicted examples to all of the dataset's instances. It is provided by:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

$$Accuracy = \frac{\mathrm{TP + TN}}{\mathrm{TP + TN + FP + FN}} \qquad (13)$$

Accuracy measures the overall correctness of the model by calculating the proportion of true results (both true positives and true negatives) among the total number of instances. The CNN model achieved an accuracy of 98.47%, while the SSACNN model yielded a notably higher accuracy of 99.16%. This significant improvement in accuracy reflects the SSACNN model's enhanced ability to classify instances correctly, demonstrating its potential as a robust classifier for this specific application.

F-Measure

A harmonic mean of precision and recall is called an F-measure.

$$f = 2 * \left(\frac{P*R}{P+R}\right) \qquad (14)$$

The F-measure, or F1-score, provides a harmonic mean of precision and recall, offering a balanced assessment of a model's performance. It is especially useful when there is an uneven class distribution. In the experiments, the CNN model exhibited an F1-score of 98.33%, whereas the SSACNN model achieved a higher F1-score of 99.16%.

This further validates the superiority of the SSACNN model in balancing both precision and recall, ensuring consistent classification accuracy across different classes.

Table 3. Performance Evaluation result for CNN and SSACNN

Models	Precision	Recall	F-measure	Accuracy
CNN	98.39	98.57	98.33	98.47
SSACNN	99.18	99.28	99.16	99.16

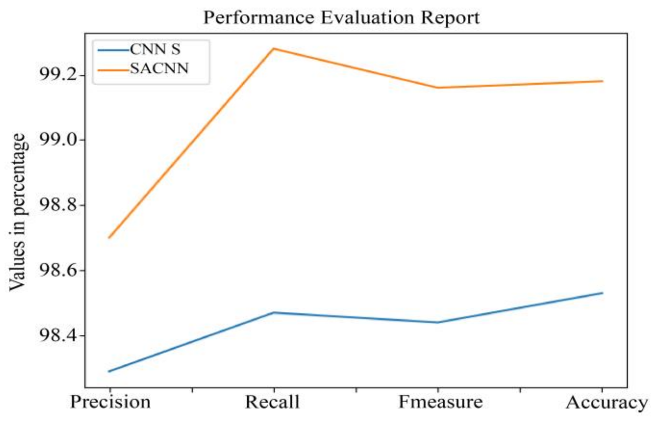

Fig. 4 Performance metrics value comparison report

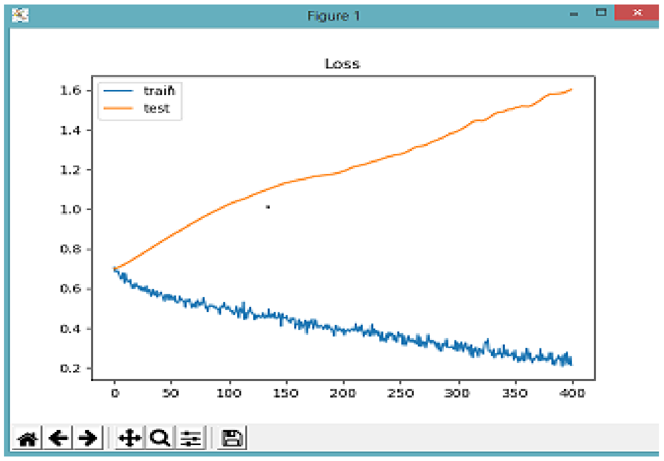

Fig. 6 SSACNN train and test loss

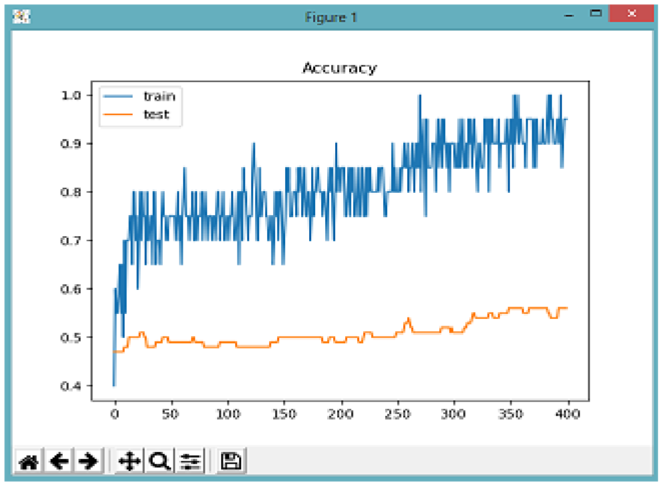

Fig. 5 SSACNN train and test accuracy

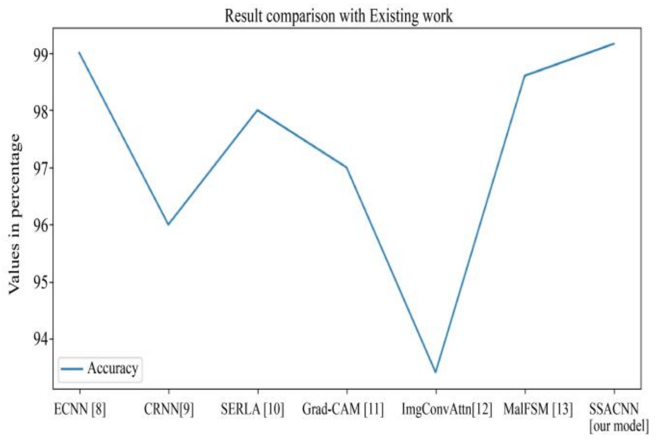

Fig. 7 Accuracy comparison with Existing works

A preliminary CNN model is constructed to classify malware issues, utilizing the Adam optimizer and categorical cross-entropy loss function across 100 epochs. Figures 3 depict the class distribution of the BIG15 dataset. Figure 4 depicts the accuracy and loss for the test and training datasets, respectively, over each epoch. Table 3 gives the performance evaluation results for CNN and SSACNN. Likewise, the proposed SSACNN was trained and evaluated using identical parameter settings, resulting in the accuracy and loss values depicted in Figures 5 and 6.

Six models' evaluation performances are shown, along with comparative data, using the Microsoft BIG15 dataset. As can be seen, the suggested model had the best accuracy, scoring 99.16%. Figure 7 shows the accuracy comparison with prior works. It is challenging to assess the effectiveness of deep learning algorithms based solely on classification accuracy. Therefore, the link between the anticipated and actual values of the classifier is divided using a confusion matrix known as an error matrix to define the performance index, which is used to measure the performance of the classifier.

4.2.3. Discussion

The SSACNN (Squirrel Search Algorithm Convolutional Neural Network) model has demonstrated remarkable effectiveness in detecting and classifying IoT malware. Its superiority over traditional CNN models is evident from the significantly improved performance metrics, which reflect the advantages of incorporating the Squirrel Search Algorithm (SSA) for weight initialization and optimization. This section outlines key reasons why the SSACNN model is highly effective for IoT malware detection.

1. Enhanced Detection Accuracy: The SSACNN model achieves a notable accuracy of 99.16%, which is higher than the conventional CNN's accuracy of 98.47%. This improvement underscores the effectiveness of the SSA in optimizing the CNN’s weights and convolutional layers, which leads to better identification and classification of malware patterns in IoT environments. Given the evolving and sophisticated nature of IoT malware, such high accuracy is crucial for minimizing false alarms and missed detections.

2. Reduction in False Positives and False Negatives: Precision and recall are two critical measures in malware detection, as they reflect the model’s ability to avoid false positives and false negatives, respectively. The SSACNN model achieved a precision of 99.18%, which signifies that it accurately identifies malware without wrongly classifying benign IoT traffic as malicious. Additionally, the high recall value of 99.28% shows that the model can effectively detect malware present in the dataset, reducing the risk of missing actual threats. This balanced performance is key in IoT environments, where the trade-off between over-detection and under-detection is sensitive.

3. Improved Generalization with Optimal Weight Initialization: One of the core challenges in training deep learning models like CNNs is the initialization of weights. In the traditional CNN approach, improper weight initialization may lead to the model being trapped in local optima, limiting its generalization capability. The SSACNN model addresses this issue by using the SSA to explore and find better initial weight configurations. By doing so, it ensures that the CNN begins with a set of weights that are closer to the global optimum, thereby improving its ability to generalize across various types of malware, especially in highly diverse and dynamic IoT environments.

4. Efficient Training and Reduced Overfitting: The utilization of SSA alongside the back-propagation method also contributes to the overall efficiency of the model’s training process. While back-propagation effectively tunes the weights to minimize error, SSA optimizes the initial weights, allowing the model to converge faster and with fewer epochs. Furthermore, the combination of SSA and CNN mitigates overfitting—a common problem in IoT malware detection—by ensuring that the model does not become overly biased toward the training data. This makes SSACNN more robust when applied to unseen malware samples.

5. Adaptability to Complex and Evolving IoT Threats: The complex and constantly evolving nature of IoT malware poses a significant challenge for traditional detection systems. The SSA’s global search capability allows the SSACNN model to adapt more effectively to variations in malware patterns, even as new threats emerge. The dynamic optimization process of SSA ensures that the CNN is well-prepared to handle the wide diversity of malware types and their continuous evolution in IoT ecosystems.

6. Scalability for Large IoT Networks: Given the vast number of connected devices in IoT environments, scalability is a crucial factor for any malware detection system. The SSACNN model, with its improved initialization and reduced training complexity, is well-suited for deployment across large-scale IoT networks. It can process and analyze vast amounts of data in real-time, making it an efficient solution for monitoring and securing IoT devices against potential malware attacks.

5. Conclusion

The research presents a unique hybrid framework for malware classification that integrates computer vision and deep learning, aiming to rectify the deficiencies in current malware variant classification methods. The framework includes the classification of distinguishing characteristics derived from malware binaries in image format, and features are extracted using a convolutional further classified with the proposed SSACNN classifier. The present model achieved 99.16% in detecting the IoT malwares on the Microsoft malware dataset. The results of experiments comparing the model's performance with those of conventional CNN and SSACNN further demonstrate the model's strong detection performance.

In order to further enhance the effectiveness of the detection component, feature extraction and classification algorithms are refined with transfer learning to forecast more unlabeled data.

Acknowledgements

Formal analysis and Investigation: VSJ; Resources: VSJ; Methodology: VSJ; Writing Original Manuscript: VSJandKN;Supervision:KN.

References

[1] Lionel Sujay Vailshery, Number of Internet of Things (IoT) Connections Worldwide from 2022 to 2023, with Forecasts from 2024 to 2033, Statista, 2024. [Online]. Available: https://www.statista.com/statistics/802690/ worldwide-connected-devices-by-access-technology/

[2] Zhiwei Guo et al., "Robust Spammer Detection Using Collaborative Neural Network in Internet-of-Things Applications," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9549-9558, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[3] New IoT-Malware Grew Three-Fold in H1 2018, Kaspersky, 2018. [Online]. Available: https://www.kaspersky.com/about/press-releases/new-iot-malware-grew-three-fold-in-h1-2018

[4] Baoguo Yuan et al., "IoT Malware Classification Based on Lightweight Convolutional Neural Networks," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3770-3783, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[5] Danish Vasan et al., "IMCFN: Image-Based Malware Classification Using Fine-Tuned Convolutional Neural Network Architecture," *Computer Networks*, vol. 171, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[6] Daniel Gibert, Carles Mateu, and Jordi Planes, "HYDRA: A Multimodal Deep Learning Framework for Malware Classification," *Computers & Security*, vol. 95, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[7] Jueun Jeon, Jong Hyuk Park, and Young-Sik Jeong, "Dynamic Analysis for IoT Malware Detection with Convolution Neural Network Model," *IEEE Access*, vol. 8, pp. 96899-96911, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[8] Danish Vasan et al., "Image-Based Malware Classification Using Ensemble of CNN Architectures (IMCEC)," *Computers & Security*, vol. 92, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[9] Seungho Jeon, and Jongsub Moon, "Malware-Detection Method with a Convolutional Recurrent Neural Network Using Opcode Sequences," *Information Sciences*, vol. 535, pp. 1-15, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[10] Yifei Jian et al., "A Novel Framework for Image-Based Malware Detection with a Deep Neural Network," *Computers & Security*, vol. 109, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[11] Giacomo Iadarola et al., "Towards an Interpretable Deep Learning Model for Mobile Malware Detection and Family Identification," *Computers & Security*, vol. 105, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[12] Qikai Lu et al., "Self-Attentive Models for Real-Time Malware Classification," *IEEE Access*, vol. 10, pp. 95970-95985, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] Zixiao Kong et al., "MalFSM: Feature Subset Selection Method for Malware Family Classification," *Chinese Journal of Electronics*, vol. 32, no. 1, pp. 26-38, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[14] Microsoft Malware Classification Challenge (BIG 2015), Kaggle, 2015. [Online]. Available: https://www.kaggle.com/c/malware-classification/

[15] Nadia Daoudi et al., "DexRay: A Simple, Yet Effective Deep Learning Approach to Android Malware Detection Based on Image Representation of Bytecode," *Second International Workshop: Deployable Machine Learning for Security Defense*, pp. 81-106, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[16] V.N. Ganapathi Raju et al., "Study the Influence of Normalization/Transformation Process on the Accuracy of Supervised Classification," *2020 Third International Conference on Smart Systems and Inventive Technology*, Tirunelveli, India, pp. 729-735, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[17] Mohit Jain, Vijander Singh, and Asha Rani, "A Novel Nature-Inspired Algorithm for Optimization: Squirrel Search Algorithm," *Swarm and Evolutionary Computation*, vol. 44, pp. 148-175, 2019. [CrossRef] [Google Scholar] [Publisher Link]