NATURALISTA CAMPANO
ISSN: 1827-7160
Volume 28 Issue 1, 2024

A Novel Cyber Security Malware Detection Using Optimization Based Fine-Tuning Dual-Channel Convolutional Neural Network (DCCNN)With Spider Monkey Optimization (SMO) And Hierarchical Particle Swarm Optimization (HPSO)

P.Vijayalakshmi[1*], Dr. D. Karthika[2]

[1*]Research Scholar, P.K.R Arts College for Women, Gobichettipalayam, Tamil Nadu, India.
Email: [1*]vijuswaamy@gmail.com
[2]Associate Professor & Head, School of Computer Science, VET Institute of Arts and Science (Co-education) College, Erode, Tamil Nadu, India.
Email: [2]karthikad@vetias.ac.in

Abstract: The implementation of conventional IoTs (Internet of Things) concepts to industrial sectors as well as apps is known as the Industrial Internet of Things (IIoT). Smart cities, smart grids, linked automobiles, smart homes, and supply chain management are just a few areas where IIoT is being used. Cybercriminals, however, are increasingly focusing on those networks. It has been demonstrated that conventional static techniques for IoT malware detection and evaluation are inadequate for comprehending the behaviour of IoT malware in order to avoid and mitigate its effects. The fields of DL (Deep Learning) and BDA (Big Data Analytics) have enormous promise for creating strong security protocols for IIoT networks. This research presents a hybrid DL strategy to identify files contaminated with malware on an IoT network.In order to detect malware, a hybrid DCCNN (Dual-Channel Convolutional Neural Network) named Hybrid SMO-HPSO, which combines Spider Monkey Optimisation (SMO) and Hierarchical Particle Swarm Optimisation (HPSO) is suggested. CB-STM-RENet is a DCCNN Fine-Tuning (FT) technique that uses the STM concept in a novel way. To address this, researchers have put forth a novel convolutional block STM that can perform region-basededge-based and region-basedprocedures both independently and in tandem. Investigating intensity inhomogeneity, boundary-defining features, and region homogeneity is made easier by the methodical application of edge and region algorithms in conjunction with convolutional processes. I.T. The visual representation of unpacked binary files for both malicious and safe apps is the malware dataset utilised for this investigation.This dataset is a result of the IoT malware detection Google Code Jam (GCJ). According to the outcomes of the research, the suggested solution outperforms current approaches by Acc, R, specificity, P, MCC, F1-score, and an average time for processing for each malware classification when it comes to measuring cyber security risks in the IoTs.

Keywords: Cyber Security, DM (Data Mining), IoTs (Internet of Things), DCCNN (Dual-channel Convolutional Neural Network), HPSO (Hierarchical Particle Swarm Optimization), SMO (Spider Monkey Optimization), Malware detection

1. Introduction

The current rapid proliferations of malware have increased cyber security risks. According to Symantec internet security threat report [1] which states that in the past three years, the amount of novel malware types has exceeded one billion. The use of packing and encryption by hackers to produce novel malware strains is growing. Therefore, it is important study to identify malware variants quickly and accurately so that cyber security experts can grasp their harmfulness and other characteristics [2]. Cyber threats are having a growing influence on the goodwill and stakeholder confidence of public and commercial institutional systems of data. Cyberattacks breach corporate IT systems to cause problems, entertainment, or commercial disruptions. Experts

engage in cyber challenges where servers or websites goes down [3].Contemporary cyber threats and hackers demand payment from the affected organisation in order to restore service, availability, or webpages.

Software intended to harm a system of computers, smart devices, or Internet-connected device is known as malware. It usually has the intention of stealing information or interfering with a network. One of these potentially harmful threats that is included into IoT gadgets is malware, which is very difficult for detection. Malicious software is only functional on the platform that it was designed for. Executable ELF files on Linux systems are similar in functionalities to .exe,.obj,.dll, etc. (PE-Portable Execution files) of Windows based systems. Though most IoT gadget CPU configurations differ, they execute on modified versions (ARM, MIPS, ×86, etc.) of Debian Linux distributions. Zero-day malware have been on the rise very recently making it complex to individually analyse their attacks. Because of this, developing a malware detection approach is essential for threats directed directly against IoT devices and infrastructure [4].

In order to detect prospective threats, majority of malware detections rely on signatures for identifications where incoming traffics are compared to identified malware stored previously [5]. Even though these methods are incredibly accurate and successful in identifying known attacks, they usually fall short when it comes to identifying fresh threats or attacks without signatures [5]. They also require a significant amount of manual labour and resources in order to update attack signatures [6]. They are therefore unable to identify zero-day threats. Much effort has gone into addressing these limitations, with several approaches focusing on detecting anomalies or behavioural research [7]. A graphical illustration of the network traffic was employed in multiple studies [8] to identify malware problems. Nonetheless, the literature has two primary drawbacks. First off, the efficiency of the suggested solution was evaluated solely using the accuracy metric as an evaluation metric. To evaluate the false as well as positive alarm rates, accuracy itself is insufficient. Next, DL models were applied and demonstrated their potential to boost performance [9]. However, this occurs at a significant cost in computation that could not be noticeable or viable for IoT-based systems having constrained processing power.

The suggested model employed conventional ML (Machine Learning) to categorise transformed PCAP into images into normal or abnormal classes, though the usage of TL (Transfer Learning) for classifying images are widely used. Conventional ML is used since TL has drawbacks when it comes to image classification [10]. One drawback is the phenomenon referred to as a negative transfer, which happens in Tloutcomes, reduce the accuracy or performance of new models. TL works best when both models' starting and target difficulty are equivalent. If the new task utilises a set of training data that differs significantly from the previous run, the trained models may not perform properly. Even when programmers think two training bits of information are the identical, algorithms might nonetheless differ.Since there are no rules governing which actions are related to each other or how algorithms decide what to do, negative transfer is difficult to address. Finding the best AI models with TL is another challenge. The trainable parameters of the dense layers will also change if the initial layers are eliminated. Furthermore, beginning to remove layers from dense ones may be a good idea, but calculating how many layers to remove in order to avoid overfitting takes time. Overfitting inherently limits any predictive approaches. There is significant data bias. Overfitting occurs when a new model absorbs qualities and noise from training data, affecting its performance. Nowadays, there are two main types of techniques used for malware recognition: static feature-based and dynamic feature-based.Static malware detection examines or disassembles the malware's essential data to determine its operation routines, file structure, and other characteristics.

As these networks expand, malware is using the internet to target nodes, laptops, and smartphones, making attacks on networked IoTs easier. Numerous methods, both dynamic and static, were developed for identifying malware via Windows. Code executions in simulated situations are used by dynamic techniques to identify malware characteristics [11]. Examining function calls, investigating parameters or data flows, following directions, or doing visual examinations of codes can all be used to spot malicious conduct. Techniques for learning modify the number of neurons in layers where large mistakes lead to poor classifier performance by using the error rates of the activation function.This study suggests using a hybrid type of DL technique to identify files contaminated with malware on an IoT network. For malware detection, a DCCNN named Hybrid SMO-HPSOis suggested.

2. Related Works

An improved Faster-RCNN (Region-CNN) is reported in [12], where TL is used for malware detections with code texture analysis. In order to identify malware programmes, the researchers of their study utilised code visualisation to show suspicious behaviours. The researchers employ Region Proposal Network (RPN) to identify the salient characteristics of those textures and CNN to extract specific malware texture visual analysis aspects in order to effectively counter code obfuscation. Faster RCNN subsequently classified malware for quick convergences. In [13], three potential approaches to categorising malware programmes according to various file formats are discussed and examined: Three approaches are available for classifying malware: (a) an ensemble that uses a combination of features extracted using SVM (Support Vector Machine) or LR (Logistic Regression) and RNN-based methods for identifying malware assembly files; and (c) CNN-based method that uses structures such as AlexNet, ResNet, and VGG-16 to identify malware produced files after they are rendered as images. The primary advantage of using LR or SVM ML models was their ability to combine sequential and graphical methodologies, resulting in improved efficiency and decreased memory use.

The study in [14] examined the possibility of using CNN and VGG16 models for classifying malware into nine distinct families. First, Hexadecimal byte representations of altered malware files were included into the decimal equivalents of the original malware datasets. The RGB values of each picture pixel were then represented by grouping these decimal numbers into three distinct groups. After that, the pixels were transformed into picture files. The CNN and VGG16 models were utilised to process the photos. Each of the five test series delivered was graded using a specific set of performance markers. The experiments demonstrated that using the optimised VGG16 model, which is DL, produced superior performance results than CNN.

Aslan et al. [15] proposed a novel hybrid architecture for effective integration of pre-trained networks. The Malimg, BIG 2015, and Malevis datasets were used to train and assess the DNNs developed in the study. The study's findings revealed that their proposed technique performed better than competing ways for correctly categorising malware with 97.78% accuracy rates on Malimg datasets—a higher accuracy rate than previous malware detection techniques that make use of MLTs. Bozkir et al. [16] examined contemporary CNNs (Resnet, Inception, DenseNet, VGG, and AlexNet) combine paradigm generation and inference time. It was also recommended to use a unique malware data set consisting of 8750 training and 3644 test cases from 25 different classes. DenseNet networks achieved an ideal accuracy of 96.48% using 3-channel (RGB) vision. Ullah et al. [17] employed a hybrid technique in which TFDNNs identified pirated applications and DLTs detected malware and pirated software in networked IoTs. Next, plagiarism in source code is detected using DL. Utilising colour images, the DCNN is also capable of identifying dangerous viruses in IoT networks. The results of the testing indicate that the suggested method for identifying IoT cyber security risks outperforms current practices. According to the previous analysis, there are certain advantages and disadvantages to the current method. Detecting unknown malware is a major problem.

3. Proposed Methodology

The three primary stages of the suggested approach are data preparation, malware detection with STM-RENet, and fusion with EL (Ensemble Learning) techniques. During preparation stages, dataset was divided into training, validation, and testing sets on conversion from PE files to RGB images.

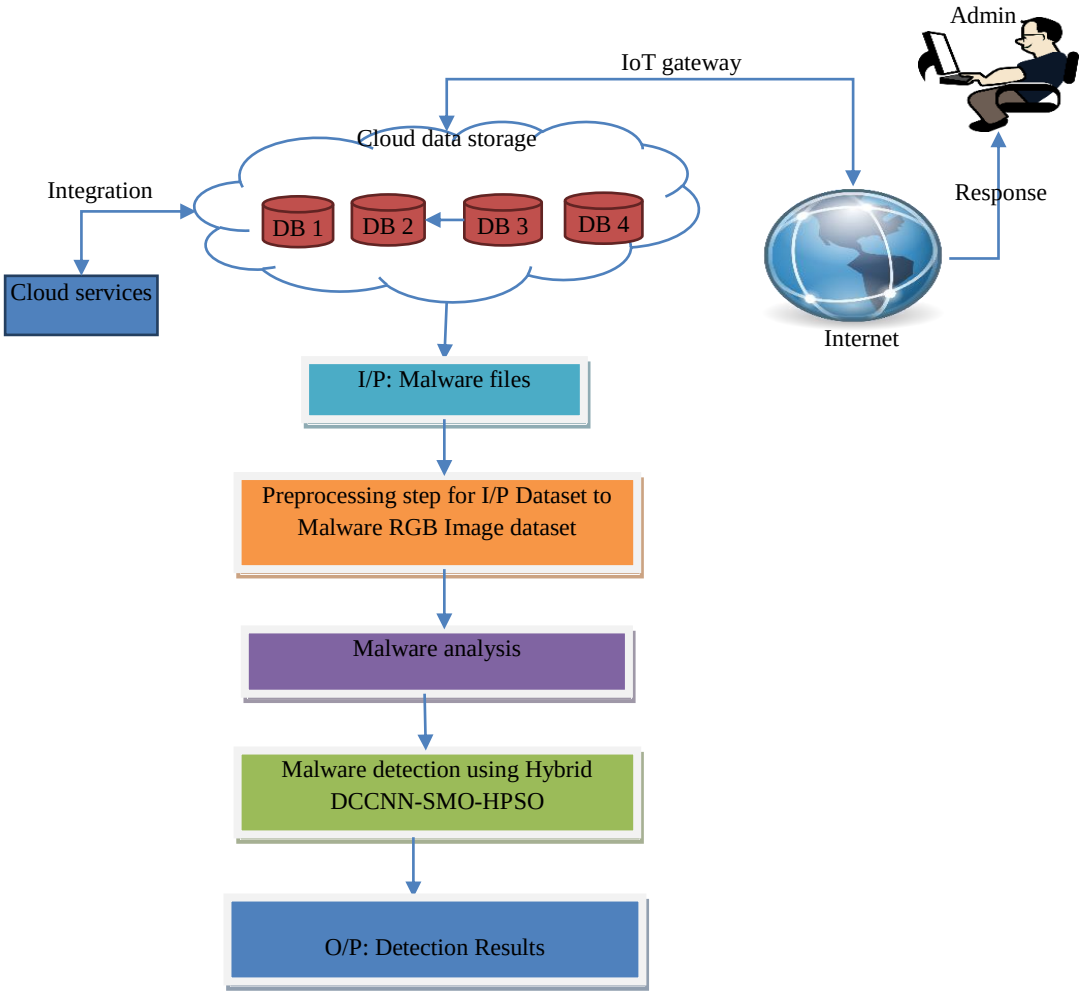


Figure 1: Proposed Methodology Diagram

Three different pre-training DCCNN architectures with pre-learned weights, modify malware dataset in subsequent stages. In order to detect malware, a DCCNN with SMO and HPSO, known as hybrid SMO-HPSO, is suggested in this study. Additionally, a novel way to (FT) DCCNN is called STM-RENet, which uses the Split-Transform-Merge concept.

a. Data Pre-Processing Phase

During this stage, the GCJ, PE files were used. To represent the output image, consecutive 3 bytes in binary files are transformed into singular pixels. Three different channels are used to display the encrypted first, second, and third bytes: RGB channels. Algorithm 1 provides a summary of this conversion procedure.

Algorithm 1: Conversion of PE files to RGB images

1: Inputs: Binary files, dimensions of Images.

2: Output: RGB images.

3: Begin

4: img = new RGB Images in specific dimensions

5: pixels = load img, row = 0, column = −1

6: while read binary file is True: do

7: bytes = read 3 bytes from the binary file

8: column += 1, row += 1

9: color = [bytes[0], bytes[1], bytes [2]]

10: pixels [column, row] = tuple(color)

11: end while

12: save img in PNG formats

13: Return RGB images

b. Hybrid DCCNN with SMO and HPSO for malware detections

For malware detection, the Hybrid DCCNN with HPSO and SMO is suggested. These networks all have 3 FC (Fully Connected) layers and 3 convolutional layers. All training sets are used to train CNN0, and the following are the parameters for each layer: Layer 1 includes 10×10 convolution kernels which apply convolutional processes to input images. At first, the procedure length is configured at 4. Down sampling is performed using 3 × 3 max pooling windows with a step length of two. Layer 2 has 40 5 x 5 convolutional kernels that perform FM (Feature Map) convolutional operations. Phase durations are commonly set to be two. Down samples are performed using 3 × 3 max pooling windows with a step length of two. Convolution kernels (60 3x3) are Layer 3, which use input feature maps to execute convolutions. Step lengths are configured to be 1. There are 3 FC layers left. A dropout layer for complete connections are employed to avoid overfitting . This implies that when the maintain pro (percentage) parameter is set to 0.5, half of the neurons in each FC layer participate in the process. Twenty output nodes are present. We chose the ReLU activation function for all activations in this work because it is expression-independent and unaffected by the VGP (vanishing gradient issue), allowing the model's convergence ratio to be maintained over time. Following the trial, the learning rate (LR) was set to 0.001.

CNN1 is trained on the low-frequency training sets, and parameters for all layer are: Layer 1 consists of 20, convolutional process by 12×12 convolution kernels executed on the input images. The stage length is primarily fixed as 2. Then, stage length 4 of the 5×5 max pooling windows are employed for down sampling. Layer 2 consists of 40, convolutional process executed by 5×5 convolution kernels on the input FMs. The stage length is basically fixed as 1. Subsequently, For down sampling, two of the four × four max pooling windows' stage lengths are used.Layer 3 executes convolution operations using the input feature maps and consists of sixty 4×4 convolution kernels. The step length is set at one. Down sampling is performed using 4 × 4 max pooling windows with a stage duration of two. CNN0's parameters are applied to the FC layers, which comprise the final three levels.

Utilising dedicated training channels, the algorithm raises the low-frequency samples' training weight. Sample equalisation is originally accomplished by processing the training samples individually. Throughout the labelling process, both channels collaborate to determine the ultimate labelling output. The low-frequency channel's features are more suited to recognise low-frequency samples. since it is exclusively trained using low-

frequency data. This lessens the labelling consequence of training with an inadequate quantity of low-frequency samples.

Proposed STM-RENet based DCCCN

Deep CNNs' robust mining of patterns capabilities have led to their widespread use in IP (Image Processing) systems [19]. The target MIA (MedicalImage Analysis) states that CNN uses the convolution technique to dynamically extract feature hierarchies by taking advantage of the image's structural data. The application of novel concepts in CNN architecture has led to a rise in their usage in tasks involving pattern recognition, classification, and detection of clinical images [20].

In this study, a unique STM and RE-based FE (Feature Extraction) processes serve as the foundation for a new Trojan CNN architecture [21]. By methodically implementing region- and edge-based procedures, RENet is able to collect border features and region homogeneity of the malware image dataset at different levels.

There are three sub-branch inside the planned block. At every branch, the idea of RE-based feature extraction is methodically used, combining the convolutional operation using max-pooling and average to provide high-level discriminating feature capture. Boundary patterns, textural variety, and geographical homogeneity are examples of distinguishing characteristics. For every convolutional execution, there are 64, 128 and 256 FM, or output channels, accordingly.

The convolution process is applied in Equation (1). x stands for input and the resulting channel.The dimensions of the channel and filter are $\boldsymbol{x}.M \times Nandp \times q$, respectively. The average (x^{avg}) and max-pooling (x^{max}) procedures are illustrated in equations (2) and (3), where's' stands for stride size. The RGB dataset is divided into three branches so that the STM-RENet may mine the patterns from it. Using an operator based on RE, it learns the typical bounds of the region-specific variations. Lastly, it uses a concatenation operation to combine the output from many pathways, perhaps capturing textural variance. To extract a variety of abstract level characteristics, two STM blocks with the identical topology are layered one after the other in STM-RENet.Equation (1) applies the convolution technique.xstands for the input channel and the output channel. The dimensions of the channel and filter are$M \times Nandp \times q$, accordingly. The average($\boldsymbol{x}^{avg}$) and max-pooling ($\boldsymbol{x}^{max}$)procedures are illustrated in equations (2) and (3), where, '*s*'stands for stride size. The RGB dataset is divided into three categories so that the STM-RENet can mine the patterns from it. Employing a method based on RE, it identifies the typical bounds of the specific- region variation. Lastly, it uses a concatenation procedure to combine the output from many pathways, perhaps capturing textural variance. 2 In STM-RENet, a sequence of STM blocks with comparable topologies are built to extract various abstract level features. The STM-RENet can extract a large variety of FM variations from the input with the use of this approach.

$$X_{m,n} = \sum_{a=1}^{p} \sum_{b=1}^{q} X_{m+a-1,n+b-1}\, f_{a,b} \quad (1)$$

$$X_{m,n}^{avg} = \frac{1}{w^2} \sum_{a=1}^{s} \sum_{b=1}^{s} X_{m+a-1,n+b-1} \quad (2)$$

$$X_{m,n}^{max} = max_{a=1,\ldots,s,b=1,\ldots s} X_{m+a-1,n+b-1} \quad (3)$$

The Proposed Deep Channel Boosted STM-RENet (CB-STM-RENet)

Because of the wide variations in the images in the information, a strong CNN algorithm is necessary for effective discriminating. By utilising CB (Channel Boosting), the suggested STM-RENet's discriminating power is increased. By taking into account several data forms through various sources, the concept of CB aids in the solution of complicated issues [22]. To enhance STM-RENet's performance, the suggested method uses CB, which creates extra feature channels using TL 2 pre-trained networks.

a. Significance of Using TL

TL is a subset of ML that makes it possible to apply the understanding of previous methods to novel problems. While there are many ways to use TL for various purposes, the most popular methods which gain knowledge in TL are based on: (1) instances, (2) features, (3) parameter exploitations, and (4) relative knowledge [23].

Work involving pattern recognition and image classification are frequently solved with feature space-TL. By adjusting network layers or introducing new layers in accordance with the target domain process, pre-training structure is modified for the target domain. Another term for this is domain adaptation. For the purpose of completing MI (Medical Imaging) operations, (Supervised Domain Adaptation) SDA-based TL using pre-trained deep CNNs has gained significant traction. By FT to the target job, this can assist give a usable collection of descriptors for features acquired from the source domain that can be applied successfully in a target domain. This lessens the need for calibration (hyper-parameter selection), which is especially challenging for deep CNNs due to the wide range of hyper-parameters and lengthy training periods.

b. Significance of Using Auxiliary Channels

CNNs with diverse architectural layouts can learn features in various ways. Multi-level data is displayed via multiple channels that were learned from various deep CNNs. Different patterns are represented by those channels, which could aid in providing a detailed explanation for class-specific traits. Combining different-level abstractions that have been acquired from various sources could enhance both the local and global depiction of the image. An individual learner makes the final choice by looking at many image-specific patterns in an adaptive feature-space-based ensemble, which is conceptualised as concatenations of original and auxiliary channels [24].

c. Proposed Channel Boosted Architectural Design

In this study, we leveraged two distinct pre-trained deep CNNs to apply SDA-TL. We define these two refined deep CNNs as auxiliary learners-1 and -2. Enhancing the suggested CB-STM-RENet model's discriminative capacity is the aim of CB.

$$C_{Boosted} = h(C_{STM} \| C_{Aux\ 1} \| C_{Aux\ 2}) \quad (4)$$

Whereas TL-FT auxiliary learner 1 and 2 give the auxiliary channels CAux1 and CAux2, respectively, CSTM displays the STM-RENet original channels in Equation 4. h(.) concatenates the original STM-RENet channels with the auxiliary channels to create the CB, which serves as the classifier's input. Figure 3 shows that the enhanced channels are assigned to convolutional block E. At the end of convolutional block E, GAP (Global-Average Pooling) is used to lower connection intensity. Finally, fully linked layers are employed to maintain conspicuous characteristics, whereas dropout layers are used to decrease overfitting.

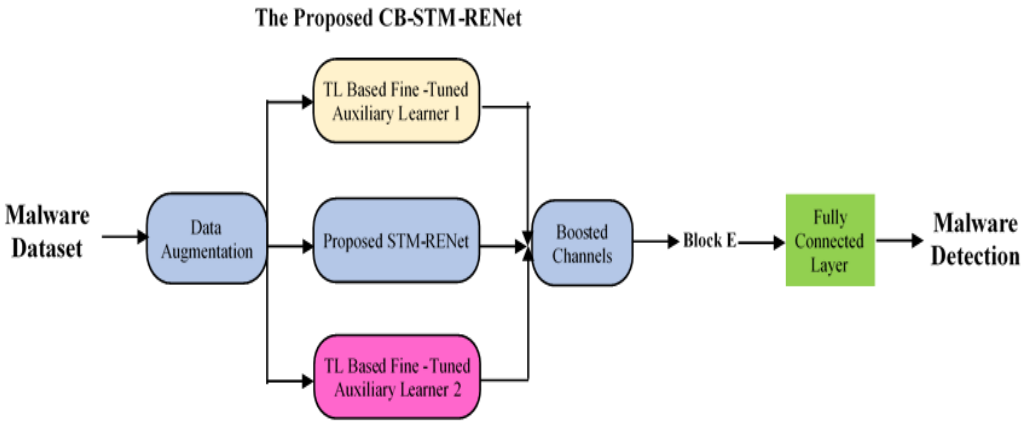


Figure 2: Boosted channels are provided to convolutional block E

Spider Monkey Optimization Algorithm

SMO is a metaheuristic search approach that leverages the fission and fission swarm intelligence strategy, drawing inspiration from the social behaviour of spider monkeys. The SMO approach is used by a tiny subset of SMs that are inspired by the resourceful foraging practices of other SMs. The period for fusion is determined by the fission, which has been developed for a single group with fewer monkeys. The system depends on how a set of features is socially organised by the dominant group, which decides either to split or merge in order to find food. Global leaders are those who control entire groups, and local leaders are those who control each individual group.

Particle Swarm Optimization

Fish schooling and bird flocking are two basic and main behaviours that encourage social activity. Food is searched for using dispersed information, or it is positioned according to the location of the discovered food. The agents travel from one location to another, searching for the food and controlling it by searching feature subsets from position nearer to them.. The food is searched from one place to another and beneficial information are delivered; consequently, it is located. The term "particle" refers to the behaviour of creatures that are optimised due to a swarm or mob.The PSO method locates every partner's position from the crowd and looks for the available space worldwide. It is then modified using Equations (5) and (6).

$v_i^{k+1} = v_i^k + c_1 r_1 (p_{best} - x_i^k) + c_2 r_2 (g_{best} - x_i^k)$ (5)
$x_i^{k+1} = x_i^k + v_i^{k+1}$ (6)
Where $p_{best} - x_i^{k+1}$Implies distances between current and best positions,$g_{best} - x_i^k$ represents distances between current and global best positions, upcoming positions of particles are represented by x_i^{k+1} while their upcoming velocities for ith particles for kth iterations are represented by $v_i^k . r_1 and r_2$are random variables with ranges between 0 and 1. The positive constants are$c_1$andc_2.

Proposed Hybrid Optimization Algorithm
The heuristic algorithm has many hybridization variations. Low-level and high-level relay or evolutionary strategies, which show the heterogeneous of homogeneous procedures employed to acquire the answer, are the two fundamental contrasts. Potential fixes for SM will be indicated by the SMO algorithm. The SMO is usually broken down into six steps: Global Leader (GL), Local Leader (LL), Leader Learning, Global Leader Decision (GLD), and Global Leader Learning (GLL).
Population Initialization
Initialised and distributed SMO of theP population as 50, the it can be denoted asSM_p (where $p = 1, 2 \ldots P$). The pth monkey belongs to theSM_p population, and its M dimension vectors are shared by the other monkeys. As theSM_p obtains the solution, those vectors or the parameters, solve the issue. Consequently, Equation (7) expresses the problem choice that the initialised SM_p is expressed as follows:
$SM_{pq} = SM_{minq} + \mho R(0,1) \times (SM_{maxq} - SM_{minq})$ (7)
where,
The pth SM's qth dimensions are denoted by SM_{pq}.
SM pins in qth direction's lower and upper bounds are$SM_{minq}$and SM_{maxq},here q =1, 2... M.
A uniformly distributed arbitrary number within the range [0, 1] is denoted by UR (0,1).
Local Leader Phase (LLP)
SM changes where it is now located. In LLP, by leveraging the existing experiences of the local leader and group members. The SM's location is only updated if the new location has a greater fitness value than the prior one. Equation (8) gives the position update for the pth SM of the lth local group.
$SMnew_{pq} = SM_{pq} + UR(0,1) \times (LL_{lq} - SM_{pq}) + UR(-1,1) \times (SM_{rq} - SM_{pq})$ (8)
Here,
LL_{lq}indicates qth dimensions of lth local GL locations.
For lth SM in lth local groups that are arbitrarily designated, SM_{rq}represents qth dimensions, provided r≠p.
Global Leader Phase (GLP)
GLP begins following LLP. The positions of SM are updated based on local group member and GL experiences. Equation (9) provides the location update formula:
$SMnew_{pq} = SM_{pq} + UR(0,1) \times (GL_{lq} - SM_{pq}) + UR(-1,1) \times (SM_{rq} - SM_{pq})$ (9)
In the qth dimension, the GL location can be represented as GL_{lq}, and (q=1, 2, 3,... M) is an index that is chosen at random.
SM's fitness is employed in this step to determine the probability prbp. This updates the location ofSM_p according to the probability value. More opportunities to improve themselves are available to candidates in better locations. The probability estimate provided by equation (10):
$prb_p = \frac{fn_p}{\sum_{p=1}^{N} fn_p}$ (10)
Here the pth SM's fitness value is fn_p. Additionally, the fitnesses of SMs' new positions are computed and compared to their former locations and highest fitness values are selected.
Global Leader Learning (GLL) Phase
The GL location is updated using the Greedy Selection (GS) approach. The position of the SM, who has the greatest fitness value in the population, is adjusted to match the position of the GL. The world leader gets to choose the best position. In the event that there are no more updates, the Global Limit Count is increased by 1.
Local Leader Learning (LLL) Phase
GS methods are applied in local groups to update LL locations. LL locations are updated using SM locations with best fitnesses in local groups. Optimum locations are assigned to LL. If no further updates are encountered, increments of 1 are added to LocalLimitCount.
Local Leader Decision (LLD) Phase

In the event that a LL fails to update their location inside a set LL Limit, every candidate in this local group will alter their positions at random in accordance with stage one or by using historical data from both LL and GL, according to the pr using Equation (11).

$$SMnew_{pq} = SM_{pq} + UR(0,1) \times \left(GL_{lq} - SM_{pq}\right) + UR(-1,1) \times (SM_{rq} - LL_{pq}) \quad (11)$$

Global Leader Decision (GLD) Phase

Because of the GL Limit, the GL position is not updated. Groups are formed from populations according to the decision made by the GL. Splitting is carried out up to the groups till a maximum amount of groups is attained. The newly created group is shaped by choosing a LL once every ten repetitions. Groups may form GLs with a limit of one, and positions cannot be updated unless the boundary separates all members of a group from one another. The acts taken by SMs, that increase productivity and handle complication in the actual world, are the basis of the SMO population.

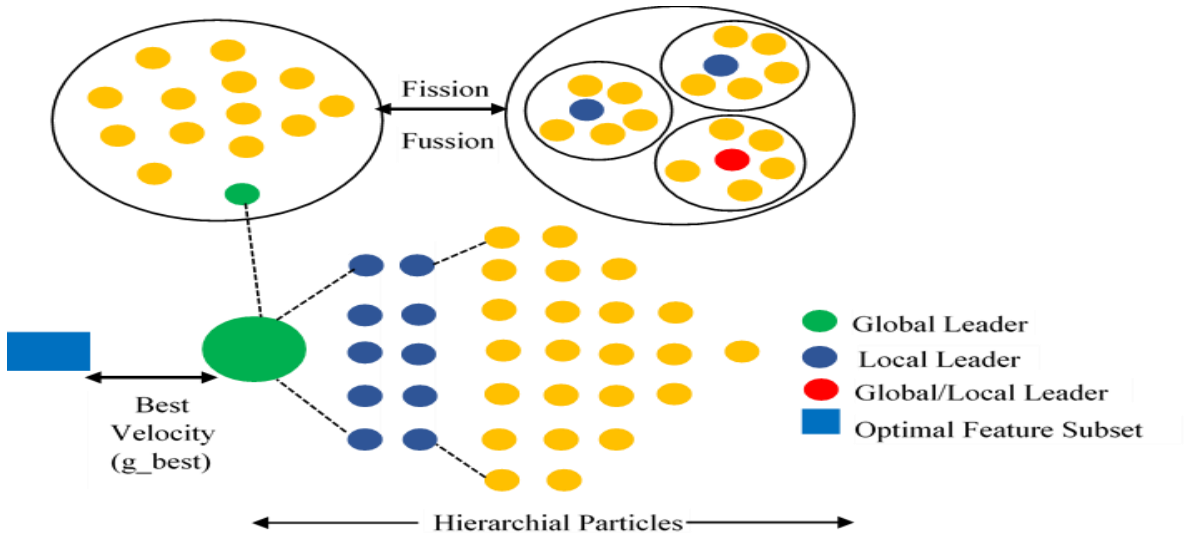


Figure 3: Suggested hybrid SMO-HPSO's diagrammatic view for g-best feature selections

Figure 4 shows a visual representation of the fitness function that an optimisation system utilises in the actual world. The feature's significance value is used to calculate the fitness value. In order to make the prediction, this gives scores for all the input feature information based on the goal variables. By using weights to calculate the feature's relevance based on node impurity, the probability of reaching the node is reduced before it does. The proportion of sampled values to the total amount of samples is used to compute the probability for the node. To determine which traits are the best, the fitness function is computed using equation (12).

$$Fitness = \frac{No.of samples\ that\ reaches\ the\ node}{Total\ number of samples} \quad (12)$$

Depending on low-level co-evolutionary features, hybrid functionalities are generated by hybridising SMO with PSO. The merge and combine variations are elements of the low-level hybrid functionality. Because the variants are used in tandem, one after the other, the co-evolutionary model is employed. The two unique variations are combined and contribute to the creation of problem-solving strategies. The hierarchical PSO can investigate SMO and generate variants with its strength based on this alteration. Equations 12, and 14 provides the updated velocity by combining the PSO and SMO variations are calculated as,

$$v_i^{k+1} = w * (v_i^k + c_1 r_1 (x_1 - x_i^k) + c_2 r_2 (x_2 - x_i^k) + c_3 x_3 (x_3 - x_i^k)) \quad (13)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \quad (14)$$

The values of the fitness function are used to find the best optimum solution. The recommended strategy takes use of the benefits afforded by the Rosenbrock function, also known as the objective function. The built-in local without approximation technique may be used with equation (15) to successfully optimise the Rosenbrock function without leveraging gradient information.

$$f(x) = \sum_{i=1}^{N-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2] \quad (15)$$

where $x = (x_1, \dots x_N) \quad \in \mathbb{R}^N$
Every variable's objective function is supplied, optimising the value. Equation (16) represents the objective function in its general form.
$g_{best} = \sum_{i=1}^{n} c_i X_i$ (16)
Herethe ith decision variable is X_i and objective function coefficients for ith variables are c_i.
The suggested Hybrid SMO-HPSO algorithm is represented by Algorithm 2.
Algorithm 2: Hybrid SMO-HPSO
1: Inputs: Malware detection sets and population sizes
2: Outputs: Optimal weights from suggested DCCNN
3: Set initial values of SM Populations, Probabilities, Parameters, Local Leader limits and Global leader limits particles with hierarchies in populations.
4: Compute importances of features.
5: Evaluate SM's fitness values.
6: Calculate global leaders and local level leader agents for feature subsets.
7: While iterations I>Max. iterations
8: For I=1to n
9: Compute fitnesses of particles using Equation (9).
10: Update particles' local best positions based on hierarchical positions.
11: based on the location update of each spider monkey's hierarchical particle velocity.
12: End for
13: Use greedy selections to select group members
14: Update SM's global optimal positions
15: For I=1to n
16: Apply Equations (5) and (6) to the local and global leader learning phase.
17: Use Equation (2) to update the particle's velocity with hierarchy.
18: Revise SM's top spot globally.
19: End for
20: End while

4. Experimental Results

The experimental analysis is implemented using MATLAB. Here, the suggested and present techniques are analysed by the GCJ dataset and a variety of assessment metrics, including MCC, F1-Score, Acc, Precision, and Recall, as indicated by Equations (17–21). Table 1 provides a description of each of these metrics in general. Additionally, AUC-PR and AUC-ROC are developed for the suggested approach. For the performance comparison, calculations are also made for TP, TN, FN, and FP

Table 1: Details of Performance Metrics

Metric Symbol	Description
Acc	Accuracy expressed as a percentage of all malwares detected
R	Recall is the percentage of malware samples that are accurately recognized as benign samples.
P	Precision measures how many malware samples were properly identified relative to all malware samples.
F1-Score	The harmonic mean of P and R is the F1-Score.
AUC-PR	Quantifies the area under Precision and Recall Curve
AUC-ROC	Quantifies the area under Receiver Operating Characteristic curve.
MCC	Mathews Correlation Coefficients
TP	Malware files have been correctly identified.
TN	Correctly identified benign files
FP	Malware files were incorrectly identified.
FN	Incorrectly identified benign files

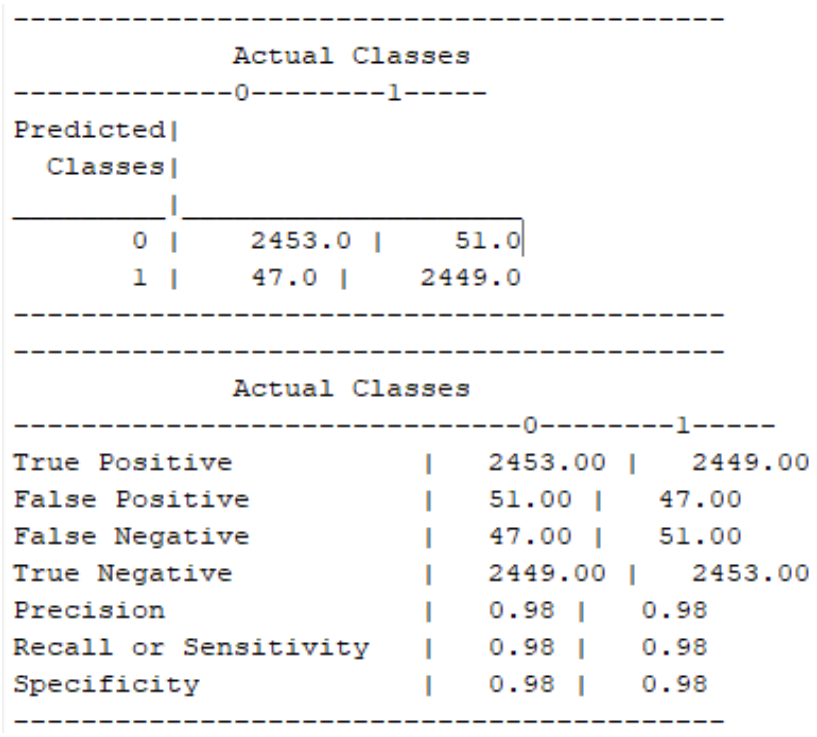
$$Acc = \frac{Predicted\ Malware\ Samples + Predicted\ Benign\ samples}{Total\ Samples} \times 100 \quad (17)$$

$$MCC = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TP+FP) * (FP+FN) * (TN+FP) * (TN+FN)}} \quad (18)$$

$$P = \frac{Predicted\ Malware\ Samples}{Predicted\ Malware\ Samples + Incorrectly\ Predicted\ Malware\ Samples} \times 100 \quad (19)$$

$$R = \frac{Predicted\ Malware\ Samples}{Total\ Malware\ Samples} \times 100 \quad (20)$$

$$F1 - Score = 2 \times \frac{P \times R}{P + R} \quad (21)$$



	Actual Classes	
Predicted Classes	0	1
0	2453.0	51.0
1	47.0	2449.0

	Actual Classes	
	0	1
True Positive	2453.00	2449.00
False Positive	51.00	47.00
False Negative	47.00	51.00
True Negative	2449.00	2453.00
Precision	0.98	0.98
Recall or Sensitivity	0.98	0.98
Specificity	0.98	0.98

Figure 4: Confusion Matrix

The suggested method's confusion matrix is represented in the figure 4.

Table 2: The comparison of classification performance among various approaches

Works	Year	Technique	F-measure (%)	Classification Accuracy (%)
GIST+SVM [25]	2011	ML	85.82	86.1
LBP+SVM [21]	2018	ML	77.49	78.05
CLGM+SVM [24]	2019	ML	91.98	92.06
DNN+SVM [25]	2019	DL	97.44	97.46
DCCNN-SMO+SVM	-	DL with optimization approaches	98.01	98.55
DCCNN-SMO-HPSO		DL with optimization approaches	98.65	98.90

Additionally, a comparison is made between the CB-STM-RENet architecture's performance with that of the current models, VGG16, inceptionv3, Resnet50,AlexNet, Shufflenet, GoolgeNet, DenseNet201,VGG19, Xception, and DenseNet201. Better results are displayed in Table 2. Utilising the Avg and Max-pooling processes, the suggested CB-STM-RENet methodically utilised region and boundary data to more effectively explore textural variance in the malware images. The features at various granularities were extracted with the aid of the channel boosting method. The suggested architecture outperformed the current approaches in terms effectiveness when the previously discussed CNN ideas were incorporated. This study used MCC, F1-Score, AUC-ROC, Acc, P, and R to quantify the importance of performance utilising DL structure.

Table 3: Comparison of Proposed Framework with the existing CNN Models

Models	Accuracy (%)	F1-Score	Precision	MCC	Recall	AUC-PR	AUC-ROC
AlexNet	92.86	0.6807	0.9960	0.5874	0.5171	0.9041	0.9685
VGG 16	94.72	0.9146	0.9552	0.839	0.8772	0.9321	0.9816
Inceptionv3	94.89	0.8055	0.9920	0.7091	0.6780	0.8972	0.9860
VGG19	95.38	0.8353	0.9902	0.7429	0.7223	0.9088	0.9739
Resnet50	95.62	0.8282	0.9971	0.7379	0.7082	0.9432	0.9848
IMDA	97.93	0.9394	0.9864	0.8796	0.8796	0.9731	0.9938
Proposed CB-STM-RENet	98.99	0.9441	0.9899	0.8801	0.8812	0.9791	0.9954

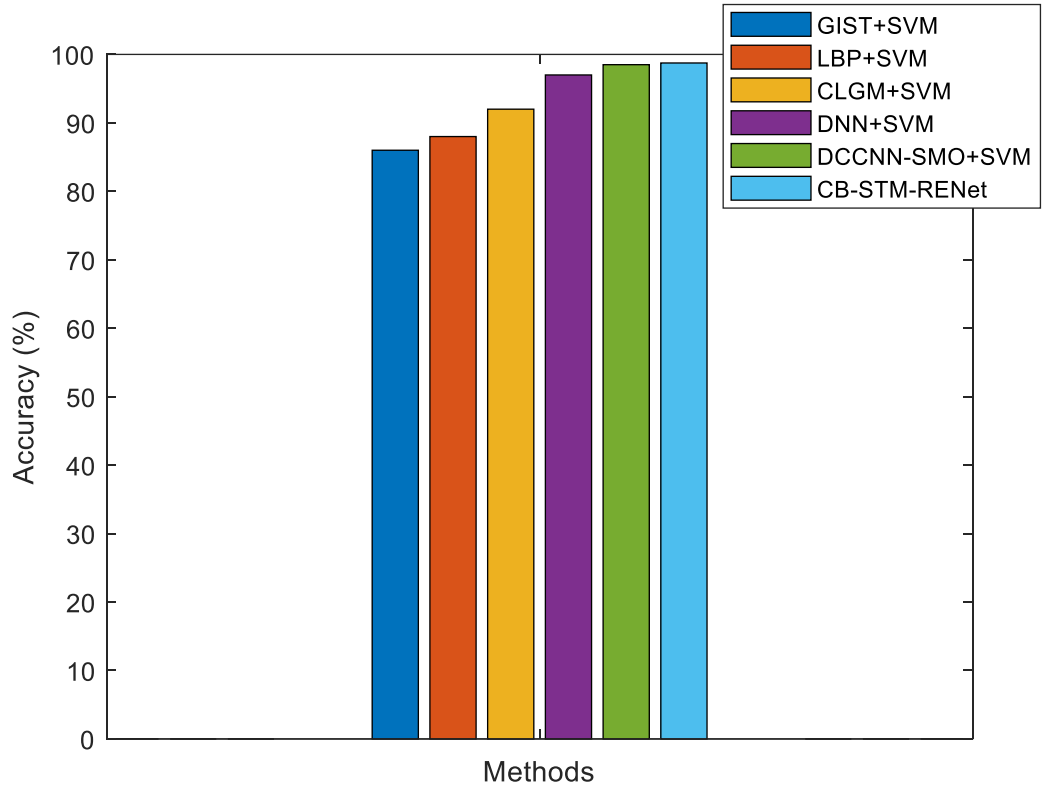


Figure 5: Accuracy Comparison

Figure 5 illustrates the manner in which the suggested CB-STM-RENet, that methodically uses region and boundary data via the Avg and Max-pooling processes, explores textural diversity in the malware images and offers a better Acc rating of 98.99%. As compared to the current techniques, inceptionv3,AlexNet, VGG16, Shufflenet, VGG19, Resnet50, DenseNet201, GoolgeNet, and Xception, the suggested CB-STM-RENet yields a higher Acc value of 98.99%.

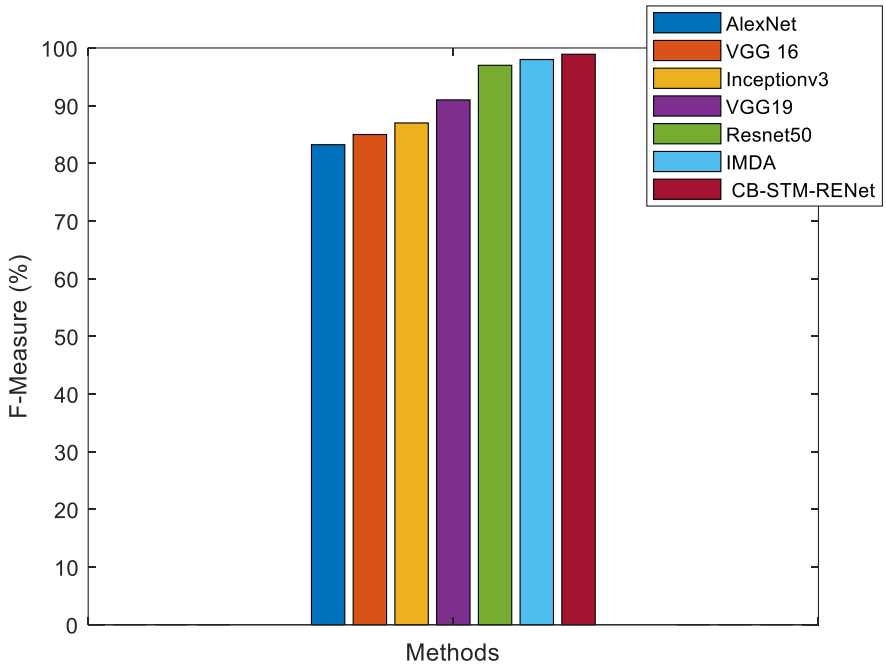


Figure 6: F-Measure Comparison

Figure 6 shows that, in comparison to the existing algorithms, inceptionv3,AlexNet,Resnet50,DenseNet201, VGG16, VGG19, Resnet50, Shufflenet,GoolgeNet, and Xception, the suggested CB-STM-RENet offers a better F-score value of 0.9441.

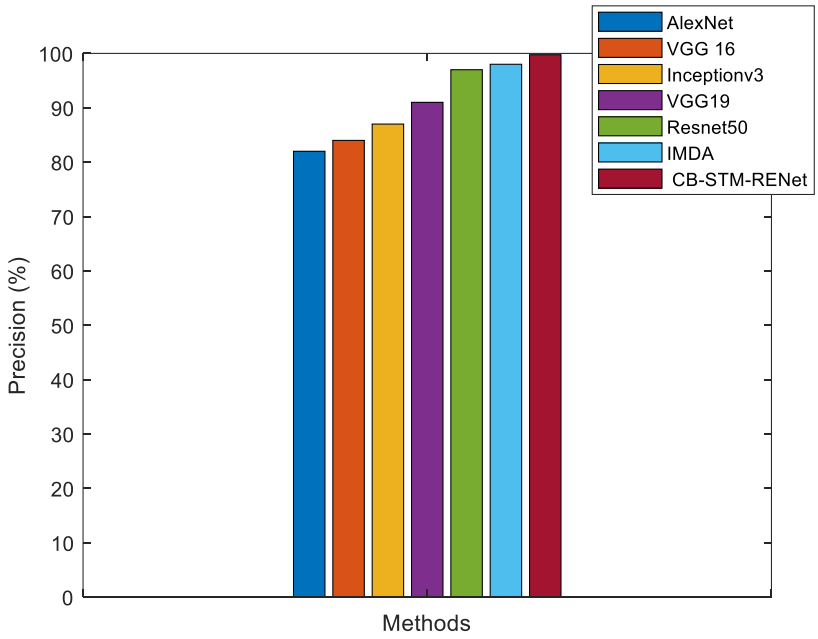


Figure 7: Precision Comparison

According to Figure 7, the suggested CB-STM-RENet outperforms the current methods, VGG19, GoolgeNet, AlexNet,inceptionv3, Xception, VGG16, Resnet50, Shufflenet, and DenseNet201with a P- value of 98.9%.

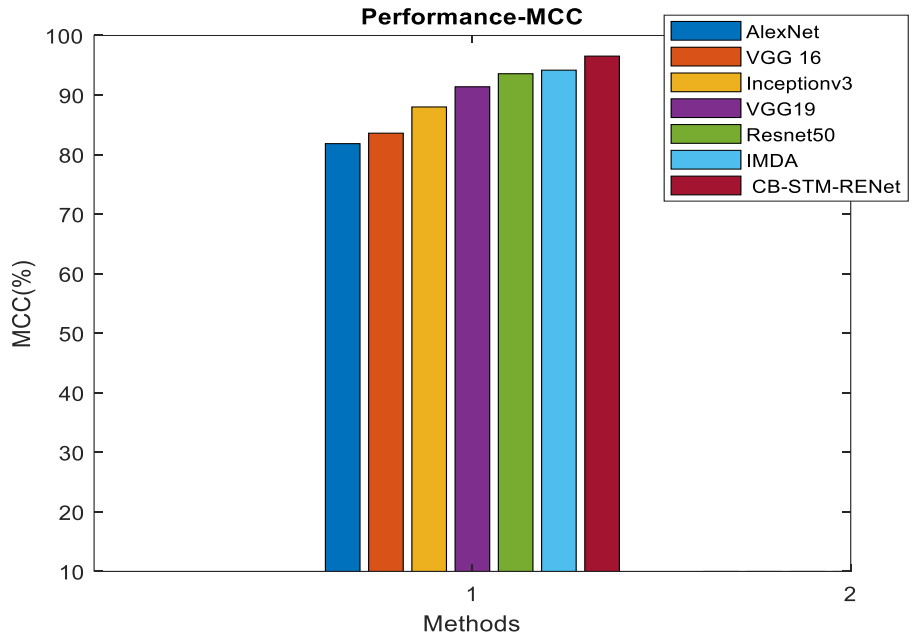


Figure 8: MCC Comparison

As can be seen in figure 8, the suggested CB-STM-RENet outperforms the current methods, GoolgeNet, Shufflenet,VGG19,inceptionv3,Xception, AlexNet, VGG16, Resnet50, DenseNet201, and Xception, achieving an MCC value of 96.1%.

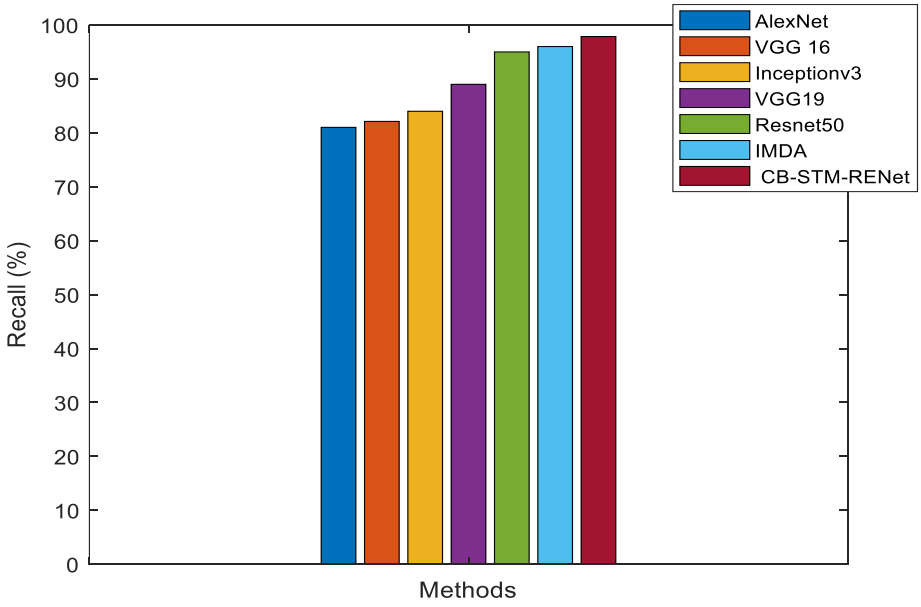


Figure 9: Recall Comparison

As seen in Figure 9, the suggested CB-STM-RENet outperforms the current methods, GoolgeNet,inceptionv3, AlexNet,Xception,Resnet50, VGG16, VGG19, Shufflenet, and DenseNet201, and with R-value of 98.12%.

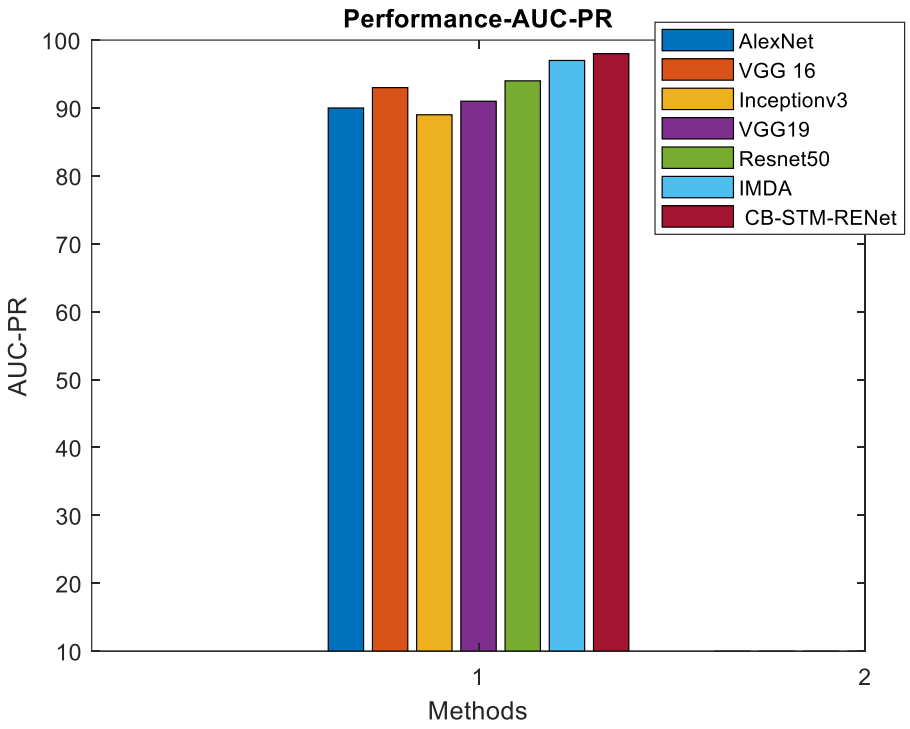


Figure 10: AUC-PR Comparison

As seen in Figure 10, the suggested CB-STM-RENet outperforms the current methods, GoolgeNet, DenseNet20, AlexNet, VGG16, Resnet50, Shufflenet,inceptionv3,VGG19, Xception, and with an AUC-PR value of 98.5%.

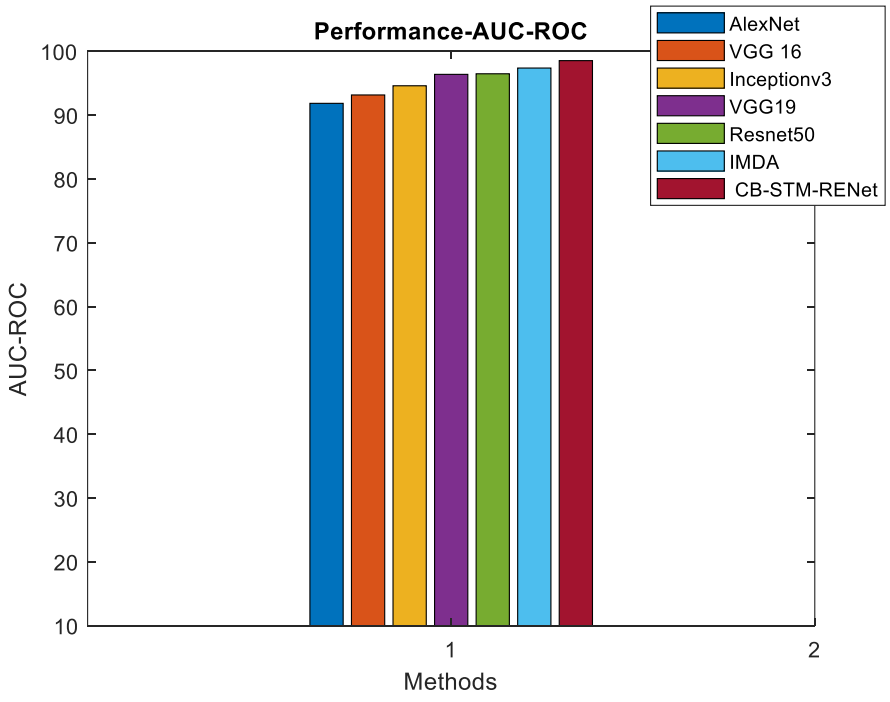


Figure 11: AUC-ROC Comparison

Figure 11 illustrates that the suggested CB-STM-RENet outperforms the current methods, GoolgeNet,AlexNet, Shufflenet,VGG16, VGG19, DenseNet201, inceptionv3, Xception, and Resnet50with an AUC-ROC value of 99.54%.

5. Conclusion

In the coming years, the IIoTs network is expanding quickly. The biggest obstacles in the realm of cyber security are finding malware risks. The integrated optimisation and DL approaches for malware file detection were presented in this study. This research presents a hybrid DL strategy to identify files contaminated with malware on an IoT network. In order to detect malware, a hybrid DCCNN) named Hybrid SMO-HPSO is suggested. CB-STM-RENet is a DCCNN -FT technique that uses the STM concept in a novel approach. In this context, a novel convolutional block STM that performs edge- and region-based procedures both independently and in tandem is proposed. According to the research outcomes, the suggested solution outperforms current approaches in terms of Acc, P, R, specificity, F1-score, MCC, and average processing duration for each malware classification when it comes to measuring cyber security risks in the IoTs.

6. References

1. Feng, S., Xiong, Z., Niyato, D., Wang, P., Wang, S. S., & Zhang, Y. (2018, December). Cyber risk management with risk aware cyber-insurance in blockchain networks. In 2018 IEEE Global Communications Conference (GLOBECOM) (pp. 1-7). IEEE.
2. Ma, X., Guo, S., Li, H., Pan, Z., Qiu, J., Ding, Y., & Chen, F. (2019). How to make attention mechanisms more practical in malware classification. IEEE Access, 7, 155270-155280.
3. Bozkir, A. S., Cankaya, A. O., & Aydos, M. (2019, April). Utilization and comparision of convolutional neural networks in malware recognition. In 2019 27th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.
4. Gaber, T.; El-Ghamry, A.; Hassanien, A.E. Injection attack detection using machine learning for smart IoT applications. Phys. Commun. 2022, 52, 101685. [Google Scholar] [CrossRef]
5. Intrusion Detection and Prevention Systems. Available online: https://bit.ly/37Bxvki (accessed on 25 November 2019).
6. Keegan, N.; Ji, S.Y.; Chaudhary, A.; Concolato, C.; Yu, B.; Jeong, D.H. A survey of cloud-based network intrusion detection analysis. Hum. Cent. Comput. Inf. Sci. 2016, 6, 19. [Google Scholar] [CrossRef][Green Version]
7. Kwon, D.; Kim, H.; Kim, J.; Suh, S.C.; Kim, I.; Kim, K.J. A survey of deep learning-based network anomaly detection. Clust. Comput. 2019, 22, 949–961.
8. Han, K.S.; Lim, J.H.; Kang, B.; Im, E.G. Malware analysis using visualized images and entropy graphs. Int. J. Inf. Secur. 2015, 14, 1–4. [Google Scholar] [CrossRef]
9. Thompson, N.C.; Greenewald, K.; Lee, K.; Manso, G.F. Deep Learning's Diminishing Returns: The Cost of Improvement is Becoming Unsustainable. IEEE Spectr. 2021, 58, 50–55.
10. Agarwal, N.; Sondhi, A.; Chopra, K.; Singh, G. Transfer Learning: Survey and Classification. In Smart Innovations in Communication and Computational Sciences; Advances in Intelligent Systems and Computing; Springer: Berlin/Heidelberg, Germany, 2021; pp. 145–155. [Google Scholar]
11. Sharma, V. K., Rizvi, S. A. M., & Hussain, S. Z. (2010). Distributed Co-ordinator Model for Optimal Utilization of Software and Piracy Prevention. International Journal of Computer Science and Security (IJCSS), 3(6), 550.
12. Zhao, Y.; Cui, W.; Geng, S.; Bo, B.; Feng, Y.; Zhang, W. A malware detection method of code texture visualization based on an improved faster RCNN combining transfer learning. IEEE Access 2020, 8, 166630–166641. [Google Scholar] [CrossRef]
13. Narayanan, B.N.; Davuluru, V.S.P. Ensemble malware classification system using deep neural networks. Electronics 2020, 9, 721
14. Olowoyo, O.; Owolawi, P. Malware classification using deep learning technique. In Proceedings of the 2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC), Kimberley, South Africa, 25–27 November 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.
15. Aslan, Ö., & YILMAZ, A. A. (2021). A New Malware Classification Framework Based on Deep Learning Algorithms. IEEE Access.

16. Bozkir, A. S., Cankaya, A. O., & Aydos, M. (2019, April). Utilization and comparision of convolutional neural networks in malware recognition. In 2019 27th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.
17. Ullah, F., Naeem, H., Jabbar, S., Khalid, S., Latif, M. A., AlTurjman, F., & Mostarda, L. (2019). Cyber security threats detection in internet of things using deep learning approach. IEEE Access, 7, 124379-124389.
18. Kingma, D.P.; Mohamed, S.; Jimenez Rezende, D.; Welling, M. Semi-supervised learning with deep generative models. Adv. Neural Inf. Process. Syst. 2014,
19. Rehman, M.U.; Shafique, A.; Khalid, S.; Driss, M.; Rubaiee, S. Future forecasting of COVID-19: A supervised learning approach. Sensors. 2021, 21, 3322.
20. Driss, M.; Almomani, I.; Ahmad, J. A federated learning framework for cyberattack detection in vehicular sensor networks. Complex Intell. Syst. 2022.
21. Bozkir, A.S.; Cankaya, A.O.; Aydos, M. Utilization and comparision of convolutional neural networks in malware recognition. In Proceedings of the 2019 27th Signal Processing and Communications Applications Conference (SIU), Sivas, Turkey, 24–26 April 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–4.
22. Shalaginov, A.; Dyrkolbotn, G.O.; Alazab, M. Review of the malware categorization in the era of changing cybethreats landscape: Common approaches, challenges and future needs. In Malware Analysis Using Artificial Intelligence and Deep Learning; Springer: Berlin/Heidelberg, Germany, 2021; pp. 71–96.
23. Roseline, S.A.; Geetha, S.; Kadry, S.; Nam, Y. Intelligent vision-based malware detection and classification using deep random forest paradigm. IEEE Access 2020, 8, 206303–206324.
24. R. Rq et al., "IoT Commercial Adoption Survey 2019 Results." [Online]. Available: https://iot.eclipse.org/community/resources/iot-surveys/assets/iot-comm-adoption-survey2019.pdf.
25. Bandara, U. and G. Wijayrathna, Detection of source code plagiarism using machine learning approach. Int J Comput Theory Eng, 2012. 4(5): p. 674