Salud, Ciencia y Tecnología - Serie de Conferencias. 2024; 3:760
doi: 10.56294/sctconf2024760

Category: STEM (Science, Technology, Engineering and Mathematics)

ORIGINAL

Ensemble classification based hybrid dual-channel convolution neural network (dccnn) with enhanced manta ray foraging optimization (emrfo) algorithm for cyber security malware threats detection

Red neural de convolución híbrida de doble canal (dccnn) basada en clasificación de ensemble con algoritmo mejorado de optimización de forraje de manta ray (emrfo) para la detección de amenazas de malware de seguridad cibernética

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

[1]Research Scholar, P.K.R Arts College for Women. Gobichettipalayam, Tamil Nadu, India.
[2]Associate Professor and Head, School of Computer Science, VET Institute of Arts and Science (Co-education) College. Erode, Tamil Nadu, India.

Cite as: Vijayalakshmi P, Karthika DD. Ensemble classification based hybrid dual-channel convolution neural network (dccnn) with enhanced manta ray foraging optimization (emrfo) algorithm for cyber security malware threats detection. Salud, Ciencia y Tecnología - Serie de Conferencias. 2024; 3:760. https://doi.org/10.56294/sctconf2024760

Submitted: 29-12-2023 **Revised:** 20-03-2024 **Accepted:** 11-05-2024 **Published:** 12-05-2024

Editor: Dr. William Castillo-González

ABSTRACT

Introduction: this study suggests usage of hybrid deep learning (DL) for identifying malwares in Internet of Things (IoT) networks. Furthermore, Channel Boost STM-RENet (CB-STM-RENet) is proposed as a DCCNN optimization technique that extends the split-change-merge model. Malware detection is performed using Hybrid Dual Channel Convolutional Neural Network (DCCNN) and Manta Ray Forage Optimization.
Methods: in this context, introduce a single-block convolutional STM known as DCCNN in CB-STM-RENet that performs local and spatial processing at the same time. The systematic use of the region and the deployment of parallel socialization processes facilitate the investigation of the unity of the region, the diversity of forces and the defining characteristics of the region. Three versions of DL: STM-RENet, DenseNet201 and InceptionResNetV2 (IRNV2) are proposed which work together to optimize DCCNN using split-change-merge in a unique way to improve generalization Hybrid learning. This dataset is a Google Code Jam (GCJ) for IoT malware detection challenges.
Results: the experimental results of the suggested method are better than existing methods for obtained accuracies and values of precision, specificity, F1 scores, MCC, and avg. processing times in classifications of cyber threats.

Keywords: Internet of Things; Data Mining; Cyber Security; Dual-Channel Convolution Neural Network (DCCNN); Malware detection; STM-RENet; DenseNet201; InceptionResNetV2 (IRNV2); Ensemble; Support Vector Machine; Classification.

RESUMEN

Introducción: este estudio sugiere el uso de aprendizaje profundo (DL) híbrido para identificar software ilegal y malware en redes de Internet de las cosas (IoT). Además, Channel Boost STM-RENet (CB-STM-RENet) se propone como una técnica de optimización DCCNN que amplía el modelo de división, cambio y fusión. La detección de malware se realiza mediante la red neuronal convolucional híbrida de doble canal (DCCNN) y la optimización del forraje de Manta Ray.
Métodos: en este contexto, introduzca un STM convolucional de un solo bloque conocido como DCCNN en CB-STM-RENet que realice procesamiento local y espacial al mismo tiempo. El uso sistemático de la región

y el despliegue de procesos de socialización paralelos facilitan la investigación de la unidad de la región, la diversidad de fuerzas y las características definitorias de la región. Se proponen tres versiones de DL: STM-RENet, DenseNet201 e InceptionResNetV2 (IRNV2) que trabajan juntas para optimizar DCCNN utilizando división-cambio-fusión de una manera única para mejorar la generalización del aprendizaje híbrido. Este conjunto de datos es un Google Code Jam (GCJ) para los desafíos de detección de malware de IoT.
Resultados: los resultados experimentales del método sugerido son mejores que los métodos existentes en cuanto a exactitudes y valores obtenidos de precisión, especificidad, puntuaciones F1, MCC y promedio. tiempos de procesamiento en las clasificaciones de ciberamenazas.

Palabras clave: Internet de las Cosas; Minería de Datos; Seguridad Cibernética; Red Neuronal Convolucional de Doble Canal (DCCNN); Detección de Malware; STM-Renet; Densenet201; Inceptionresnetv2 (IRNV2); Conjunto; Máquina de Vectores de Soporte; Clasificación.

INTRODUCTION

Cyber security is even more important due to the proliferation of computer systems where IoT encompasses the Internet, wireless networks, and smart gadgets. The practice of securing networks and computer systems from unauthorised access, theft, corruption, service disruption, or electronic data tampering. hardware or software is called "cyber security".(1) The complexity of today's communication networks is raising serious concerns about cyber security. The major focus is on systems' integrities and installations of required components. Every second, several cyber assaults are launched, including malicious attacks, reverse engineering, phishing, espionages, denial of service (DoS), Distributed denial of service (DDoS) and direct accesses.

Concerns about user privacy are growing as cloud computing becomes more and more prevalent.(2) A distributed cloud storage strategy utilising encryption was developed to avoid attacks on clouds.(3) This increased the attacker's burden, assuring dependability and enhancing privacy.(4) ML and DL approaches were used to counteract increasing IoT botnet flash assaults on social media platforms and IT sectors.(5,6) Even though IoT helped us advance, it also gave hackers a chance to steal our data and violate our privacy by using botnets to operate from far-off places. These botnets can be used to carry out any of the previously listed types of attacks. Various kinds of intrusion detection systems, or IDS, were created to look into any questionable behaviour or network traffic and notify us. With the intention of not only recognising suspicious activity but also taking the necessary steps to lessen or eliminate those threats, many writers constructed IDS employing machine and DL algorithms.(7)

In recent years, assaults of all types have increased in number and ferocity. Malware, often known as "malicious software," is a programme or piece of code that has the ability to violate security and privacy, disrupt computer networks, enable unauthorised access to information and disclose personal information. Examples of terms used for malware include ransomwares, virus, Trojans, worms, and spywares. A minimum of one element is at least affected by malware in technologies resulting in damages. It is expected that 5,22 billion i.e. 66 % of the global population will have smartphones by 2020 (Data Reportal, 2021) where 93 million used Android based mobile devices and a perennial increase of 1,8 %.(10) In fact, it has been used in a broad range of devices, including wearable's such as augmented reality headsets and smart watches, tablets, smartphones, and wearable's. Because the Android operating system is widely used, it is open-source, and we frequently save sensitive data on our mobile devices, rogue code writers write increasingly aggressive scripts every day with the intention of stealing our data.(11)

The constant conflict between security professionals and hackers has made malware into a dynamic ecosystem. The malware ecology varies annually as evident from studies. Hackers and cybercriminals never give up easy, even with all the anti-virus technologies in place, especially if there is a profit to be made from infection. Some once-popular malware variants appeared to be losing ground in 2022 when hackers proliferated vulnerabilities that were new or seldom used.(12) There are signs that hackers are using the IoT and email to spread separate malware. Large corporations and governments continue to receive more attention than average internet users, especially when it comes to ransomware outbreaks.

In 2020, there was malware activity from employee to employee in 61 % of the organisations. In 2021 and 2022, that percentage rose to 74 % and 75 %, respectively. According to the security experts at Check Point, mobile hazards affected 97 % of businesses in 2020 (Mobile Security Report).(13) Furthermore, at least one person from 46 % of firms used a dangerous mobile application. As of August 7, 2022, Google's Transparency Report(14) states that about four million browser warnings were sent to users who attempted to visit websites marked as hazardous by Safe Browsing. In a study on the growth of ransomware attacks, on August 3, 2022,(15) Statista's research department found that in the first half of 2022 there were approximately 236,1 million attacks worldwide. The damage caused by the ransomware reportedly reached five billion. $8 billion in 2017,

$8 billion in 2018, $11,5 billion in 2019, $20 billion in 2021, and $265 billion in 2031. A malware attack can have a number of major consequences, such as a sluggish computer, limited storage, unwanted programmes, network failures, losses of critical data, increases in outgoing traffics, DoS, DDoS, etc. These results indicate that, given the havoc that malware has caused, it is critically necessary to identify reliable and fast ways to identify malware and lessen its effects.

Malicious software developed and grew increasingly complex over time. The conventional approach includes the deployment of antivirus software. A common method of detection employed by antivirus software is signature-based. Signature-based identification is a very effective method for identifying a particular type of malicious software by looking for predefined byte groupings within an item. Since no fresh software signature is kept in the database, its primary drawback is its inability to identify malware that is almost zero-dated.(17) Consequently, the antivirus programme is unable to locate the novel malware signature in time to identify novel malware variations.

Behavior-based recognition was created in order to essentially address the shortcoming of the signature-based method. Rather of looking for malware signatures, it examines the behaviour of the framework and looks for any unusual patterns. The behavior-based procedure is constrained by the system's execution time and the amount of additional storage required. This technique focuses mostly on how the programme behaves when it is being executed. A programme is classified as benign if it functions normally; if not, the file is classified as malware. Examining the behavior-based approach in detail allows us to conclude that one of its main shortcomings is that it leads to a lot of incorrect classifications, considering how easily legitimate applications may be mistaken for malwares or run like regular programmes.

The two primary categories that the research community uses to categorise malware analysis technologies are static and dynamic analyses,(18) useful tool for quickly identifying malware, but additional obfuscation techniques like packaging can be used to hide it. Dynamic analysis successfully solved the shortcomings of static analysis technology, notwithstanding its susceptibility to viral evasion. Hardware characteristics can be used to get around software feature limits for tackling the malware evasion problem in dynamic analysis. As shown in Campanile et al.(19), malware may be efficiently identified by extracting hardware features from performance counters (IPC, cache behaviour, memory behaviour, etc.). Based on the above-described experiments,(20) used hardware performance counters (HPC) with unsupervised approaches for identifying malwares. However, the previously indicated method's classification accuracy was insufficient to explain malware behaviours employing HPC.

Most modern harmful software severely restricts one's capacity to deconstruct and analyse software in several ways. Obfuscation techniques are frequently used to modify malware. They entail changing the code's syntax to make it more challenging to decipher while maintaining functioning. When code optimisation features included in compilers are paired with obfuscations, reverse engineering malware becomes more difficult. The challenge of classifying malware is introduced into the image by obfuscations. Various machine learning (ML) techniques are used to classify malware, from DL models that can operate directly on raw data to models that require human characteristics before training.

One drawback of DL approaches compared to shallower models is that they are more likely to overfit data while training on datasets with less counts of samples.(21) This may be problematic in fields like programme analysis, especially when it comes to classifying malware, more efforts and resources are needed to find sufficient relevant data examples. This problem also commonly arises in other areas, such as image recognitions and classifications.(22) Insufficient training data may be easily resolved by applying certain algorithmic changes to virus images without altering their semantics and new instances can be generated from prior knowledge. This entire procedure, which is known as "data augmentation," is a crucial part of deep learning. It was initially presented in Statista Our Research and Content Philosophy(23).

Detection methods have progressed in the malware detection hierarchy. To facilitate analysis and detection, negative duplicates found in the dataset or derived from opcodes get transformed into grayscale/color images. Various image processing methods were applied in order to preserve uniformity of the images and extract features. The amalgamation of deep learning algorithms with visualisation approaches was extensively employed and demonstrated to be advantageous. When trained on malware images, DL models acquire malware features more precisely. The most often used DL models are CNN-based ones. Another approach in malware analysis is transfer learning (TL).(24) It is the application of knowledge gained from one work to another that is related yet unrelated. This implies that in ML environments, layer weights and parameters of DL trained models of collections can be applied to analyze other collections. With big data sets, TL drastically minimises the training.

Two kinds of tasks may be carried out with TL: feature extraction and classification. Generally speaking, there are two primary uses for pre-trained models: feature extraction and classification. Examples include ResNet34, ResNet50, VGG-16, ResNet101, and so on. Models that have previously undergone training to handle a certain problem are known as pre-trained models of problems can be used after changes for different tasks in place of creating new models.(25) The majority of them are CNN students.

In order to detect malware, this research paper suggested using a hybrid DCCNN with enhanced Manta

Ray Foraging Optimisation (EMRFO). In addition, a CB-STM-RENet DCCNN fine-tuning approach is given, which extends the split-transform-combination paradigm. In this context, we suggested a novel convolutional block STM, CB-STM-RENet-based DCCNN with EMRFO, that performs region- and edge-based operations independently and simultaneously. The DCCNN fine-tuning approach based on convolutional convolution theory has been distributed in three DL versions: STM-RENet, DenseNet201, and IRNV2. These updates improve accuracies and values of precision, recall, F1 scores MCC, and avg. processing times of classifiers.

Related work

The number of malware assaults on IoT devices is increasing, and old ways of detecting malware are not as effective since these algorithms leverage the traditional signatory libraries and interaction knowledge of malware researchers. However, malware may be automatically detected and adapted to any discipline using ML and DL approaches.(26,27) Four processes make up an ML-based malware detection method: building the dataset, creating features, training the model, and assessing the model. By identifying reliable and illuminating characteristics, feature engineering determines the correctness of the model and describes the A.P.K.s. The primary features that define A.P.K. s are the AndroidManifest.xml and classes. dex files which encompass basic details about A.P.K. s, including hardware specifications, filtered intents, requested permissions, and APK components. A class v. Dex is converted into a compact format made up of operands and opcodes from Dalvik commands. as well as deconstructing classrooms. The malware detection models may also be trained using dex files to gain further information, such as flow diagram controlling(28) and API dependency graph.(29) Running a programme on a certain platform provides information on its dynamic behaviour, including network activities, service openings, system calls; file operations, phone calls, and cryptographic functions.(30) When combined with static features, these dynamic features result in more accurate patterns and increase search speed.

Malware detection made extensive use of classical ML techniques like Random Forest (RF),(31) Support Vector Machines (SVM), and DL models such as CNN,(32) Long Short-Term Memory (LSTM).(33,34) Many ML and DL algorithms for IoT malware detection have demonstrated promising and reliable performance. These tools attack weaknesses in IoT apps and firmware, allowing them to infect both the whole network and its edge devices. Anti-malware apps have helped to increase the adoption of ML techniques and processing capacity in recent scientific advancements.

Bendiab et al.(35) utilised a 1000 network file (pcap) for ResNet50 pre-training to analyze IoT malicious code traffic. Parra et al.(36) suggested DL frameworks using LSTM and DCNN for detection and also reduce botnet assaults and phishing schemes in clouds. The two primary security techniques function together.(1) To identify botnet attacks, the backend runs a cloud-based ad hoc network model (LSTM). Distributed phishing attempts are identified across multiple IoT devices using CNN clustering. DCNN model detects DDoS attacks at the application and phishing layers of IoT devices. The detection of phishing attacks and the protection of the origin of IoT devices are achieved by integrating distributed CNN and ML into the customer's IoT devices. The suggested CNN plugin security model was trained on provided datasets with both phishing and phishing URLs. Backend LSTM models are trained on N_BaIoT datasets. The advantages of joint training were lower resources, communication requirements and maximum utilization of the important features learned. The system works better as a whole when multiple requests are combined using graph fusion. A proposed CNN model IoT micro security plugin gets a higher F1 score for detecting phishing attacks. The results of the study exhibited that the suggested model could detect background/ device-level attacks.

Haddadpajouh et al.(37) suggested Recurrent Neural Network (RNN) for examining malware threats in IoT. RNN is used to analyse executions of app operation codes on ARM based IoT. A combined total of 270 benign and 281 malware files make up the IoT application dataset that is used to train the suggested models. Secondly, 100 fresh IoT malware samples were evaluated with three distinct LSTM trained models (i.e. not earlier exposed to model). Based on the analysis of Tenfold cross validations, experimental result indicate accurate identifications of new malwares in second configurations using 2-layer neurons. In the end, LSTM usage delivered best results.

METHODS

The suggested approach is broken down into three primary stages: preparing the data, employing ensemble learning techniques for fusion, and applying CB-STM-RENet for malware detection. The dataset was transformed into RGB images from Portable Execution (PE) files in prepreparation stages. The second stage involved loading the pre-learned weights into the DCNN structures, fine-tuning them, and utilising the malware dataset to train them. This research suggests using Hybrid DCCNN in conjunction with Enhanced Manta Ray Foraging Optimisation (EMRFO) to detect malware. In addition, a CB-STM-RENet DCCNN fine-tuning approach is given, which extends the split-transform-combination paradigm. In this context, we suggested a novel STM convolutional block, CB-STM-RENet-based DCCNN with EMRFO, that performs region- and edge-based operations independently and simultaneously. A DCCNN fine-tuned approach based on the distributed transform-fusion idea is provided for three deep learning versions: STM-RENet, DenseNet201, and IRNV2.These variants improved accuracies, specificities and corresponding values were obtained for precision, recall, F1-scores, MCC, and avg. processing timess values during classification of malwares.

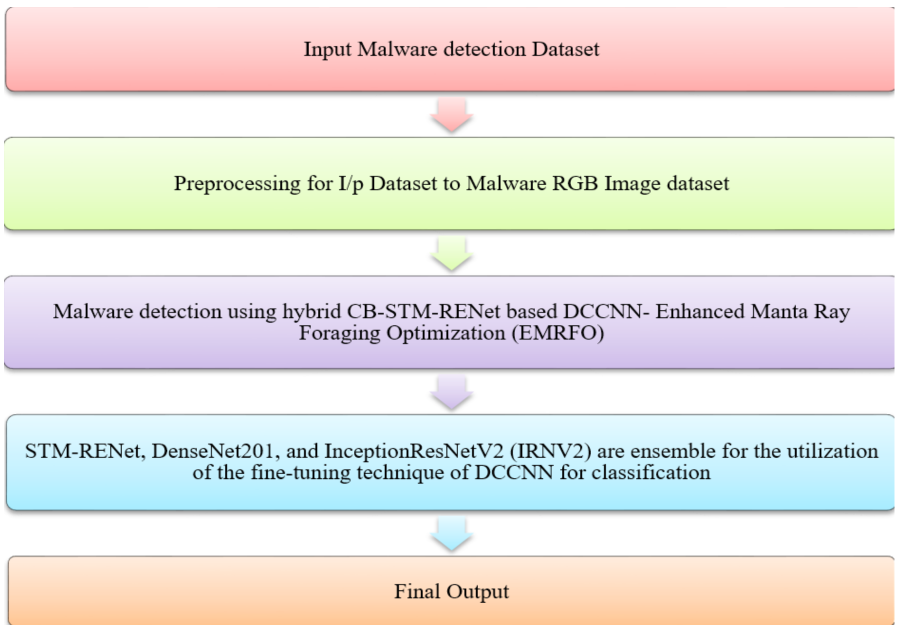

Input Malware detection Dataset
Preprocessing for I/p Dataset to Malware RGB Image dataset
Malware detection using hybrid CB-STM-RENet based DCCNN- Enhanced Manta Ray Foraging Optimization (EMRFO)
STM-RENet, DenseNet201, and InceptionResNetV2 (IRNV2) are ensemble for the utilization of the fine-tuning technique of DCCNN for classification
Final Output

Figure 1. Proposed Methodology Diagram

Data Pre-Processing Phase

In this stage, PE files recovered from broken software between 2017 and 2018 were transformed into RGB images in this step and fed into the suggested CNN models using bin2jpg technique. Output images were represented by tranforming consecutive bytes (3) into single pixels. Three different channels displayed the encrypted first, second, and third bytes i.e. red, green, and blue and saved as JPEG files.

Hybrid DCCNN with Enhanced Manta Ray Foraging Optimization (EMRFO) for malware detection

This paper builds a DCCNN model to improve recognition and accuracy. The networks had 3 convolution and connected layers. The training of CNN0 used the following: Layer 1 contained 20 10 x 10 convolution kernels that associated convolutions with inputs image features with initial step lengths of 4. The submodels used 3 × 3 binning windows and step lengths of 2. Forty 5 × 5 convolution texture layers are used to create the convolution map. 2. Step lengths were set to 2 initially. Level 3 used 60 3×3 seeds with step lengths of 1 and maximum aggregation windows of 3x3 with a step length of 2. Three levels of connection remain. To prevent these layers from overlapping, a drop coating is used. The pro hold parameter (proportion) was fixed to 0,5, i.e. half of connected layer neurons were active. There are 20 output nodes. In this study, the ReLU activation function was used for all activations because it has very good reproducibility, does not suffer from the problem of missing gradients, and thus keeps the model convergence rate constant. On testing, learning rates were set at 0,001.

CNN1 was trained with a low-frequency training set, and the following settings were used in each layer. Layer 1 consists of 20 rotating 12x12 textures that use the animation function in mixed mode. Initial step lengths were set to 2. Subsampling is performed using a 5 × 5 maximum binning windows with a step length of 4. Layer 2 is a 5 x 5 texture that applies transformations to input feature maps and using step lengths of 1. Subsampling is performed using a 4×4 maximum accumulation window with a step length of 2. The third layer is a 64 x 4 convolution kernel that applies convolutions on input feature maps using step length of 1. Subsamples use maximum 4 × 4 sampling windows using step lengths of 2. Final three layers are connected based on parameters of CNN0.

Proposed STM-RENet based DCCNN

Deep CNNs' robust pattern mining capabilities have led to their widespread use in image processing applications.[38] The goal medical image analysis states that CNN uses convolution operation to dynamically extract feature hierarchies by taking use of the image's structural information. CNNs are used more often used in categorizations, detections, and pattern recognitions of medical images in applications as a result of their creative designs.[39]

In this study, split-transform-merge block (STM) and feature extractions based on Res serve as the foundation for a new malware CNN architecture.[40] By methodically implementing region- and edge-based procedures, RENet is able to capture the malware dataset's border properties and region homogeneity at several 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 with 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 operation, there are 64, 128 and 256 output channels or feature maps, accordingly.

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

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

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

Equation (1) uses convolutions. Inputs and resultant channels are denoted by *x.M×Nandp×q* stands for dimensions of channels and filters, respectively. Equations (2), (3) show averages (xavg) and max-summing (xmax) procedures, with "s" indicating the step size. The RGB dataset is partitioned into three branches so that STM-RENet may extract patterns from them. RE-based operators assisted in learning normal boundaries of regional variances. Finally, it performs merge operations to mix outputs of many pathways for capturing texture variations. To separate distinct abstraction level functions, two STM blocks with similar topologies are placed in series in STM-RENet. This technique enables STM-RENet for extracting the various set of feature map modifications from the input.

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

A strong CNN model is necessary for effective discriminations because of wide variances in images. By utilising Channel Boosting, the suggested STM-RENet's discriminating power is increased. By taking into account several data representations from various sources, the concept of channel boosting aids in the solution of complicated issues. To enhance STM-RENet's performance, the suggested method uses Channel Boosting, which creates extra feature channels using TL from two pre-trained networks.

TL in ML allows usages of obtained knowledge by existing methods. Their common utilization involve: (1) models, (2) sites, (3) parameters, and (4) relational.

Tasks related to pattern recognition and image classification are solved by domain-based TL. The learnt architectures can be applied to target domains by altering network layers and adding new ones when needed. This is also known as territorial law. CNN, which is frequently used in medical imaging, learns from local TL. In conclusion, target features can be constructed by utilising a set of feature definitions that have been learned from source network domains and effectively applied in target domains. This reduces the need for regularisations, or additional selections, which is a challenge in deep CNNs due to the high computational costs and extended training times.

Importance of Auxiliary Channel usage

CNNs with diverse architectural layouts may learn features in different ways. Multi-level information is shown via several channels that were learned from various deep CNNs. Different patterns are represented by these channels, which might aid in providing a detailed explanation for class-specific traits. Combining different-level abstractions that have been learned from various sources might enhance both the local and global representation of the image. Spatial band recognition is characterised as a mix of base and auxiliary channels, with a single learner evaluating several visual samples before reaching a final choice.

Proposed Channel Boosted Architectural Design

This work uses dual pre-trained deep CNNs to apply supervised domain adaptation-based TL. Because of the differences in their architectural designs, these deep CNNs allow each model to extract different radiological information from chest X-rays and learn additional feature descriptors. We refer to 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 channel boosting.

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

Although CAux1 and CAux2 are auxiliary channels. The original STM RENet channels are displayed in Equation (4) after TL adjusted auxiliary learners 1 and 2. h(.) combines original STM RENet channels having subchannels for obtaining channel-boosted (Cboosted) inputs for classifications. As shown in figure 2, the E convolution block receives the amplification channel. To reduce coupling strengths, global averages at end of convolution blocks

E are used. Finally, staggered layers assist in reducing overlaps and all connected layers to preserve important properties.

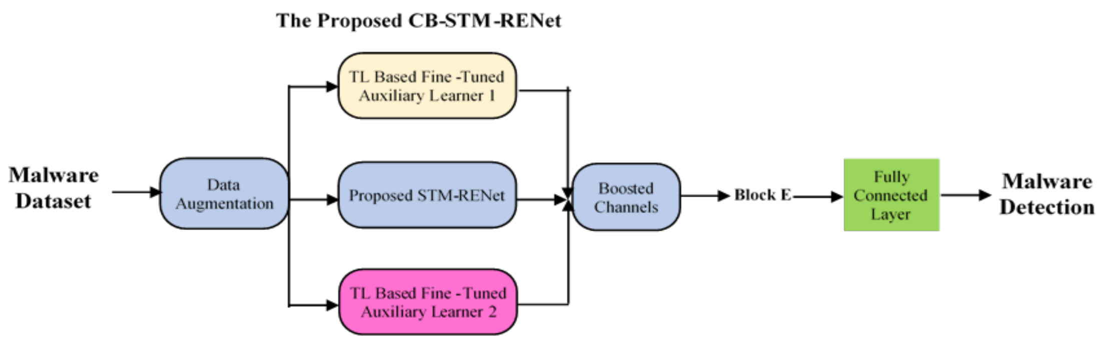


Figure 2. Boosted channels used by convolutional blocks E

Enhanced Manta Ray Foraging Optimization (EMRFO) to optimize the DCCNN

MRFO, inspired by foraging behaviours of manta rays includes chains, cyclones and somersaults, is leveraged for selecting hyper parameters in optimal manners for the CNN classifier. The optimal parameters of CNN classifiers are presumed to be the plankton having the maximum concentration of manta rays that desire to reach and feed on.

Chain Foraging: These are primary behaviours of manta rays for ideal selection of the CNN classifier parameters. Equation (5) and (6) give the mathematical derivation for these behaviours.

$$x_i(k+1) = x_i(k) + rnd_1.\left(x_{\text{best}} - x_i(k)\right) + \propto.\left(x_{\text{best}} - x_i(k)\right) i = 1, (5)$$

$$x_i(k+1) = x_i(k) + rnd_1.\left(x_{i-1}(k) - x_i(k)\right) + \propto.\left(x_{\text{best}} - x_i(k)\right) i = 2, \ldots, n \qquad (6)$$

$$\propto = 2r\sqrt{|log\ (r_1)|} \qquad (7)$$

Where $x_i(k+1)$indicate new positions of individuals, $x_i(k)$stands for current positions of ith individuals at times t, ranb are random numbers in the interval [0,1], α refers to the weight coefficient,$x_{bestspecifies}$ the plankton in terms of accuracy for optimal choice of hyper-parameters.

Cyclone Foraging: Once the manta rays get highly accurate hyper-parameters in the CNN classifier, they will travel towards the optimal parameter in chain. For the initial phase, manta rays travel randomly given by the Equations(8-9):

$$x_i(k+1) = x_{\text{rand}} + rnd_2.\left(x_{\text{rand}} - x_i(k)\right) + \beta.\left(x_{\text{rand}}(k) - x_i(k)\right) i = 1 \qquad (8)$$

$$x_i(t+1) = x_{\text{rand}} + rnd_2.\left(x_{i-1}(k) - x_i(k)\right) + \beta.\left(x_{\text{rand}}(k) - x_i(k)\right) i = 2, \ldots, n \qquad (9)$$

At subsequent stage, the manta rays will go towards the top individuals and this behavior is expressed using Equations (12-14):

$$x_i(k+1) = x_{\text{best}} + rnd_2.\left(x_{\text{best}} - x_i(k)\right) + \beta.\left(x_{\text{best}} - x_i(k)\right) i = 1 \qquad (10)$$

$$x_i(k+1) = x_{\text{best}}(k) + rnd_2.\left(x_{i-1}(k) - x_i(k)\right) + \beta.\left(x_{\text{best}} - x_i(k)\right) i = 2, \ldots, n \qquad (11)$$

$$\beta = 2e^{rd_1 \frac{\text{kmax}+1}{\text{kmax}}} Sin(2\pi r_2) \qquad (12)$$

Where *β* indicates the weight coefficient, k_{max} refers to the highest number of iterations and r_2 refers to therange of random numbers [0,1].

Somersault Foraging: In this mechanism, the food position is considered to be a pivot. Manta rays will swim in to and FRO movement surrounding the pivot and somersault to fresh optimal hyper-parameters. This way, the manta ray will revise their new location in the neighborhood till now. The mathematical definition of somersault is given as:

$$x_i(k+1) = x_i(k) + S.\left(rnd_3.x_{\text{best}} - rnd_4.x_i(\text{k})\right), \qquad (13)$$

Where rnd_3&rnd_4 indicate the random number in the range [0,1] and S stands for therangeof somersault that the manta rays can make. Genetic Algorithm (GA) is designed with the intent of simulating the biological process that takes its inspiration from the theory of evolution. The mechanism adopts the survivals of the fittest principle for fresh hyper-parameters. Subsequent, each generation computes the fittest one is computed for every individual (hyper-parameters). However, there aren't many low-quality people that survive, and there aren't many opportunities for reproduction. For the next generation, GA employs crossovers and mutations which are genetic operators that maintain genetic varieties of GA chromosomal hyperparameters in generations. The GA mutation operator randomly moves between the upper and lower boundaries. Crossovers are genetic operators that merge genetic material of hyperparameters to produce new generations. GA selects two hyperparameters at random or creates two new cases. Mutation stands for an operation that diversifies the individuals within a possible area and it is found to be effective in ensuring that all the areas was examined. For the solution to converge towards precision, crossover is crucial. The enhanced MRFO method is called the EMRFO algorithm. The inclusion of GA components, such as mutation and crossover function, modifies EMRFO. Since all of the variables used by EMRFO—such as mutation and crossover rate—are really derived from GA and are based on the iteration number, EMRFO does not require any setup parameters. Subsequently, this suggested method 1 was run 10 000 times using number of function evaluations (NFE) for fair comparisons of MRFO and GA.

```
ALGORITHM 1:  EMRFO
Step 1: Preparation
Choose specific counts of manta rays in populations (nPop) , max. iterations, and function evaluations (NFE).
Step 2: Initialization
    Set initial points of locations (hyper-parameters) of manta rays randomly x_i (k) , between possibleareas
Xi(k)=xi + rand(xu-xl)
Where xl refer to the lower limit, and xu indicates the upper limit of hyper-parameter.
Step 3: Fitness evaluation
Assess finesses fi=f(xi (k))  of manta rays at locations and get optimal solutions as xbest.
Step 4: use Foraging Model
Select a random number rnd1
If r1 lesser then k/kmax then do cyclone foraging
        Else do chain foraging
      a.  Cyclone  foraging
Select random number rnd2
If rnd2 lesser then k/kmax then shift individuals by equations (5,4-5,5)
              Else if rnd2>k/kmax, then apply equations (5,6-5,8)
      b.  Chain  foraging
Shift individuals by equations (5,1-5,3)
Assess fitness fi=f(xi(k)) , and get theoptimal solution so far as xbest
If f(xi(k+1))<f(xbest) then xbest=xi(k+1)
f(xi(k+1))<f(xbest)
     Then use somersault foraging by equation (5,9)
Use genetic operations such as mutation and crossover
Assess fitness fi=f(xi(k)), and get the optimal solution so far as xbest
Step 5: validation of stop condition
     Check if stop condition kmax is satisfied then stop, else go to step 3
End
```

In EMRFO, lower and upper bounds indicate limits of hyper-parameters forming search spaces, and best positions indicate optimal hyper-parameters and f(x) refers to function values that CNN model retrieve EMRFO algorithm. The proposed algorithm is excellent and is customized to any CNN model. The proposed technique begins with defining the lower and upper bounds and the highest number of iterations in random. Next, the EMRFO algorithm first performs the hyper-parameters initialization, which indicates the current position. Subsequently, it starts updating its position based on the earlier steps to obtain the top position (optimum hyper-parameters) having improved accuracy and reduced loss and the accuracy, loss are computed just from CNN model training.

Ensemble Modeling of DCCNN

In this research, an ensemble learning approach is suggested which integrates 3 DCCNNs structures like CB-STM-RENet, Inception ResNet V2 and DenseNet 201 for enhancing the execution by the accuracy, sensitivity, specificity, precision, and F1-score.

Inception ResNet V2

The foundation of the inception model is the scaling up of networks. To increase processing performance, the model makes use of factorised convolutions and aggressive regularisation. A hybrid Inception architecture that incorporates ResNet capabilities is the InceptionResNetV2 model. The filter concatenation stage of the Inception is replaced in this model by residual connections. By using this technique, Inception is able to preserve its processing efficiency while still reaping all of its benefits. Figure 3 shows the InceptionResNetV2 that was used in this study.

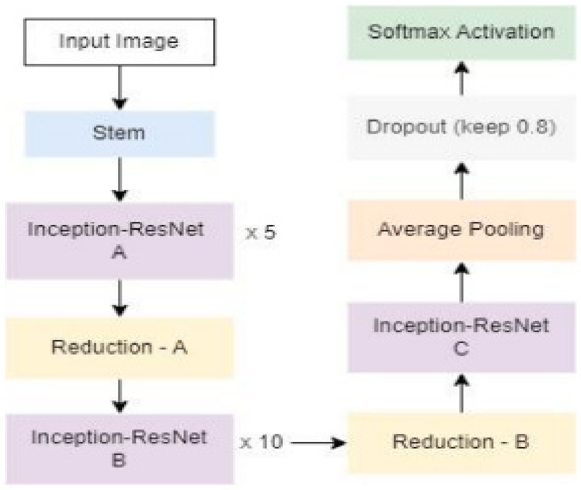


Figure 3. InceptionResNetV2 architecture

DenseNet 201

The foundation of Dense Convolutional Networks (DenseNet) is the feed-forward link between each layer. Numerous noteworthy benefits are provided by this neural networks like resolution of vanishing gradients, enhanced feature propagations, feature reuses, and greatly minimized parameters. To help with the feature map modification process caused by downsampling layers, the networks are split up into several densely linked units called dense blocks. The version of this model that was different was the dense block's network setup.

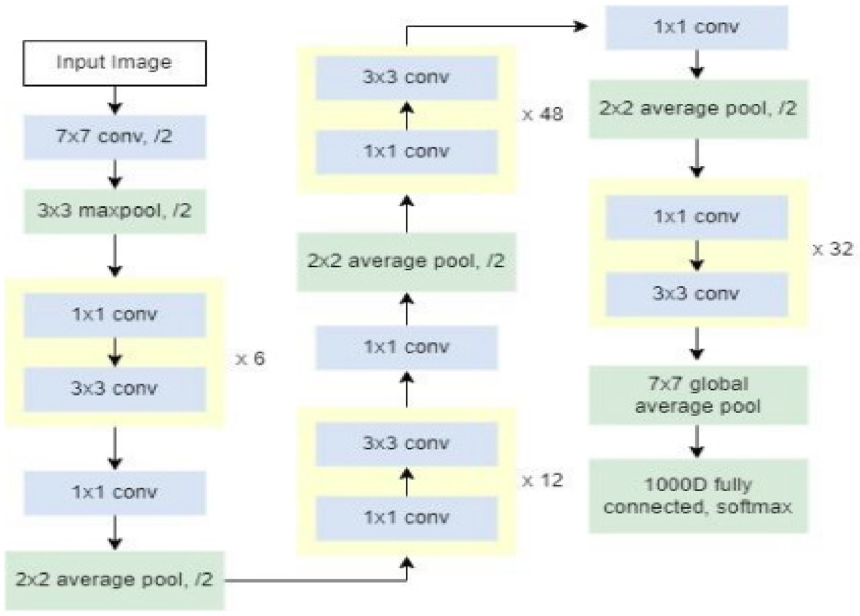


Figure 4. DenseNet201 architecture

In this study, we describe in detail the individual and ensemble performance of STM-RENet, Inception ResNet V2[41] and DenseNet 201[42] were used to classify the malware dataset and final connection layers of all three architectures have been modified and new layers have been added. The kernel module[41] has three separate widths for the convolutional layer with the grain filters (5 × 5, 3 × 3 and 1 × 1) and the ensemble layer with the 3 × 3 filters. In studies resulting from the release of the architecture Inception ResNet V2. The Inception ResNet V2 architecture is a modification of the Inception V3 model. Use DenseNet 201 in addition to STM-RENet and Inception ResNet V2. A convolutional network (DenseNet) combines two layers of the same map size. Among the many advantages of DenseNet are the reduction of the problem of missing gradients, the improvement of feature aggregation, the ease of reuse, and the significant reduction of the number of parameters. As shown in figure 3, the architecture of all CNNs has been translated into an architecture for ensemble learning.

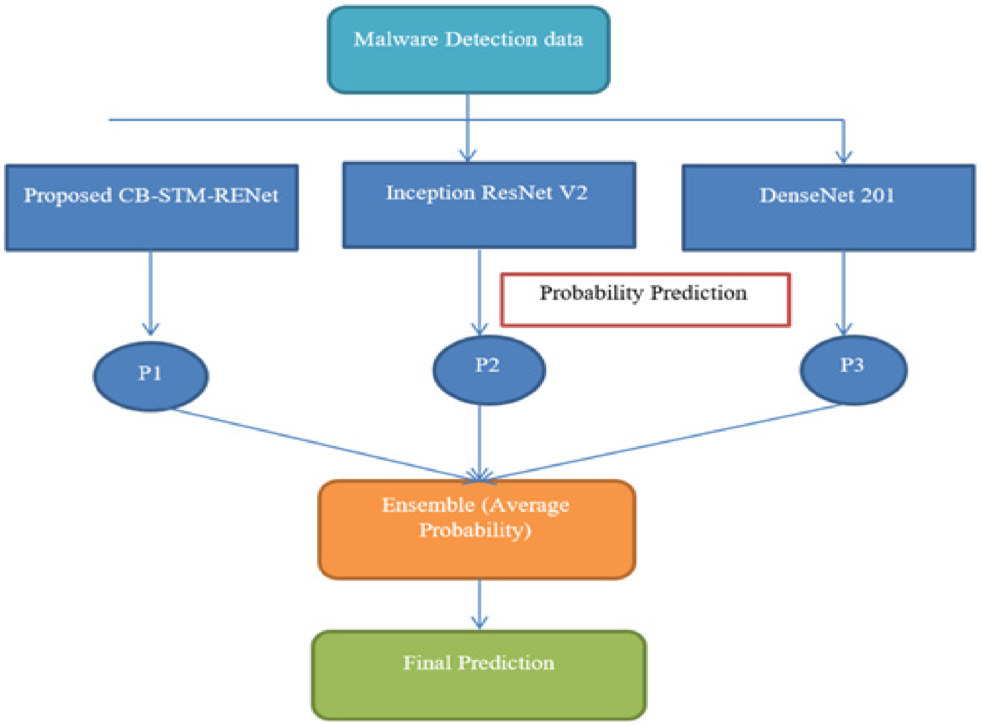


Figure 5. Proposed DCCNNs with ensemble learning by using three architecture STM-RENet, Inception ResNet V2, and DenseNet 201

The utilized structure is a hybrid learning-based strategy in which the proposed DCCNN extracts Feature maps, that can be efficiently acquired from the suggested model for malware detection in the network indicates the better performance.

RESULTS

We examine the suggested research in the case study. The Linux operating system (OS) is increasingly gaining traction among IoT devices,(43) which presents a potential target for malware developers. Linux distributes firmware and applications via the ELF file format. ELF files are binary files in two types: compressed and uncompressed binary. These are cross plates.(49) The IOT_Malware dataset utilised in this research includes a graphical representation of uncompressed binaries for malware and benign programmes.(44) This dataset is a common starting point for Kaggle IoT malware detection assignments.

Table 1. Details of Performance Metrics

Metric Symbol	Description
Acc	Accuracy as % of the total number of Malware detection
R	Recall, which is the ratio of correctly identified malware samples and benign samples
P	Precision, a ratio of correctly detected malware samples to the total malware sample
F1-Score	F1-Score is the harmonic mean of P and R.
AUC-PR	Quantifies the area under Precision and Recall Curve
AUC-ROC	Quantifies the area under Receiver Operating Characteristic curve
MCC	Mathews Correlation Coefficient
TP	Correctly Identified Malware Files
TN	Correctly Identified Benign Files
FP	Incorrectly Identified Malware Files
FN	Incorrectly Identified Benign Files

In the present research, performance evaluation metrics evaluation : Acc, P, R, F1-Score, and MCC , as presented in Eqs. (14-18) mulated for the suggested framework. True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (TN) are also computed for the performance comparison.

$$ACC = \frac{Predicted\ Malware\ Samples + Predicted\ Bengin\ samples}{Total\ Samples} \times 100 \quad (14)$$

$$MCC = \frac{(TP*TN)-(FP*FN)}{\sqrt{(TP+FP)*(FP+FN)*(TN+FP)*(TN+FN)}} \quad (15)$$

$$P = \frac{Predicted\ Malware\ Samples}{Predicted\ Malware\ Samples + Incorrectly\ Predicted\ Malware\ Samples} \times 100 \quad (16)$$

$$R = \frac{Predicted\ Malware\ Samples}{Total\ Malware\ Samples} \times 100 \quad (17)$$

$$F1 - Score = 2 \times \frac{P \times R}{P+R} \quad (18)$$

Table 2. Comparison with Malware Detection Techniques

Approaches	Source Code	Technique	Accuracy (%)
Bandara, Wijayrathna	Java	KNN	86
Cosma, G. and M. Joy	Java	LSA	99
Son, J.-W	Java	Parse Tree	90
Ullah, F	C++, Java	LSA	80
Ullah, F	C++, Java, Python	MLR	86
ULLAH	C++, Java, Python	Deep Neural Network	96
DCCNN-SMO	MATLAB	Hybrid DCCNN with Spider Monkey Optimization (SMO) called (DCCNN-SMO)	98,12
CB-STM-RENet based DCCNN- SMO-HPSO	MATLAB	Channel Boosted STM-RENet (CB-STM-RENet) with Hybrid DCCNN with SMO and Hierarchical Particle Swarm Optimization Approach called CB-STM-RENet based DCCNN- SMO-HPSO	98,9
CB-STM-RENet based DCCNN-EMRFO with ensemble Approach	MATLAB	Channel Boosted STM-RENet (CB-STM-RENet) with Hybrid DCCNN with Enhanced Manta Ray Foraging Optimization (EMRFO) called CB-STM-RENet based DCCNN- EMRFO with Ensemble model of CB-STM-RENet, Inception ResNet V2, and DenseNet 201	99,4

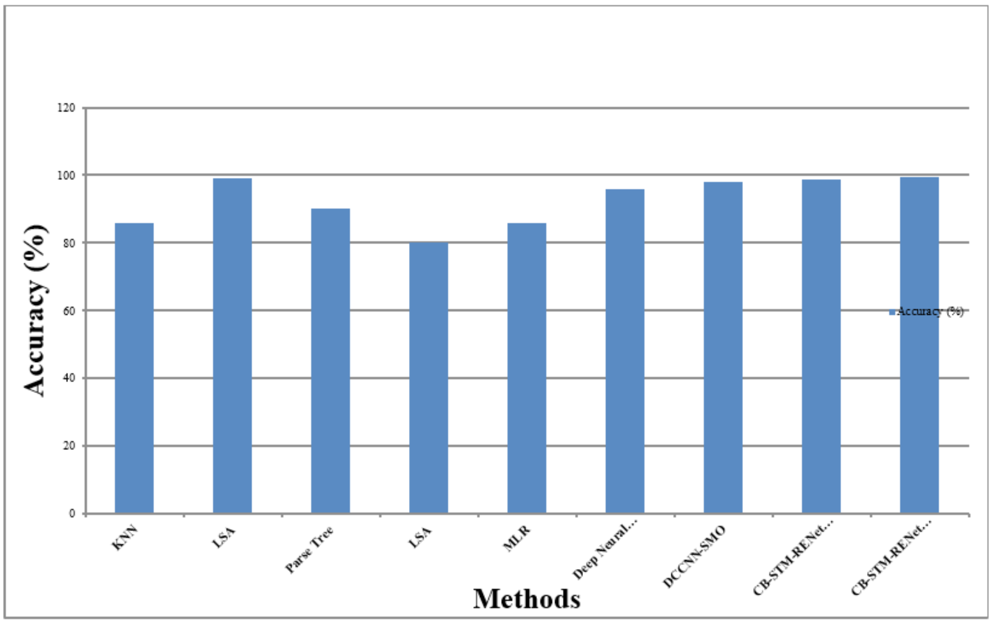


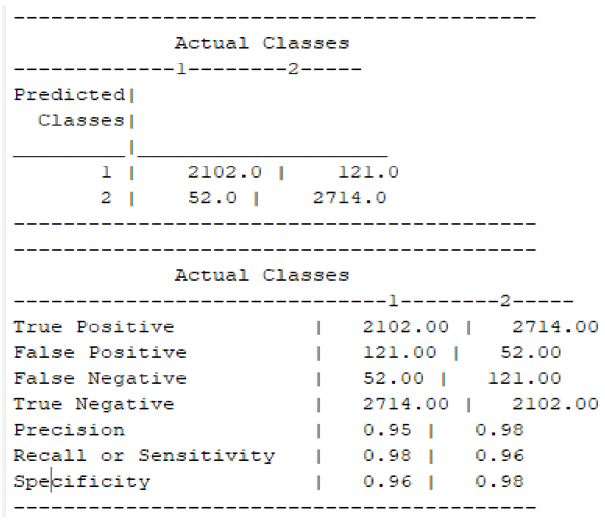
Figure 6. Accuracy Comparison of the Malware Detection Techniques

Figure 6 indicates the Acc of the Malware Detection when related with the other current methods. The suggested CB-STM-RENet based DCCNN-EMRFO with ensemble approach provides better detection accuracy of value 99,4 when compared with the other current methodologies like CB-STM-RENet based DCCNN- SMO-HPSO, DCCNN- SMO, DNN, MLR, LSA, Parse Tree and KNN.

Table 3. Comparison of Classification Performance of the Proposed Malware Detection Method for Different Image Ratio

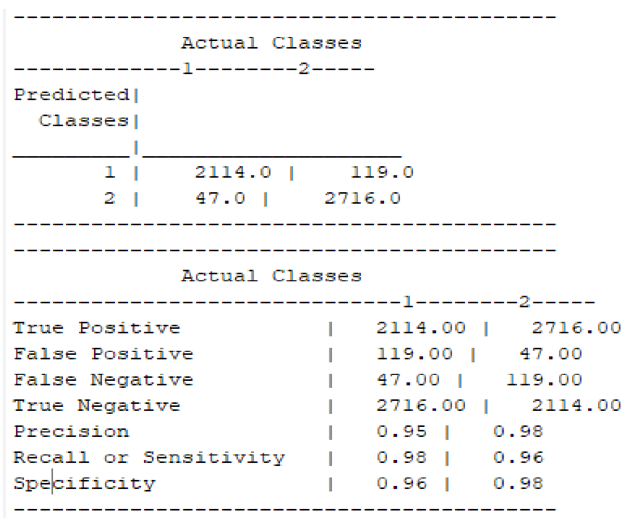
Image Ratio	Precision (%)	Recall (%)	F-measure(%)	Accuracy (%)	Time (s)
224×224	97,11	96,90	97,20	98,2	13s
229×229	98,78	99,04	98,99	99,4	20s

We also compared the performance of suggested DL-based malware detection with current ML-based malware detection researches on the Leopard Mobile dataset. These approaches used traditional image feature extraction descriptors, namely, GIST, LBP, and CLGM descriptors with SVM classifier. Table 3 indicates that the suggested malware detection technique attains superior outcomes against 3 current ML-based work by classification performance.



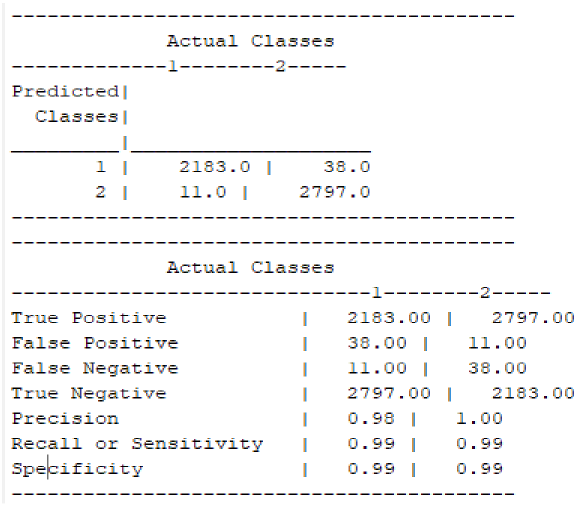
```
------------------------------------------
             Actual Classes
-------------1--------2-----
Predicted|
  Classes|
_________|_____________________
       1 |     2102.0 |     121.0
       2 |     52.0 |     2714.0
------------------------------------------
------------------------------------------
             Actual Classes
-----------------------------1--------2-----
True Positive            |   2102.00 |   2714.00
False Positive           |   121.00 |   52.00
False Negative           |   52.00 |   121.00
True Negative            |   2714.00 |   2102.00
Precision                |   0.95 |   0.98
Recall or Sensitivity    |   0.98 |   0.96
Specificity              |   0.96 |   0.98
------------------------------------------
```

(a) Confusion Matrix for densenet



```
------------------------------------------
             Actual Classes
-------------1--------2-----
Predicted|
  Classes|
_________|_____________________
       1 |     2114.0 |     119.0
       2 |     47.0 |     2716.0
------------------------------------------
------------------------------------------
             Actual Classes
-----------------------------1--------2-----
True Positive            |   2114.00 |   2716.00
False Positive           |   119.00 |   47.00
False Negative           |   47.00 |   119.00
True Negative            |   2716.00 |   2114.00
Precision                |   0.95 |   0.98
Recall or Sensitivity    |   0.98 |   0.96
Specificity              |   0.96 |   0.98
------------------------------------------
```

(b) Confusion Matrix for inceptionv2



```
------------------------------------------
             Actual Classes
-------------1--------2-----
Predicted|
  Classes|
_________|_____________________
       1 |    2183.0 |    38.0
       2 |    11.0 |    2797.0
------------------------------------------
------------------------------------------
             Actual Classes
------------------------------1--------2-----
True Positive            |  2183.00 |   2797.00
False Positive           |  38.00 |   11.00
False Negative           |  11.00 |   38.00
True Negative            |  2797.00 |   2183.00
Precision                |  0.98 |   1.00
Recall or Sensitivity    |  0.99 |   0.99
Specificity              |  0.99 |   0.99
------------------------------------------
```

(c) Confusion Matrix for Ensemble data

Figure 7. Confusion Matrix

The confusion matrix comparing the suggested and current procedures is presented in figure 7. In learning and statistics, a confusion matrix is a table that evaluates a classification algorithm's effectiveness. By displaying the percentages of TP, TN, FP, and FN predictions, it provides an overview of the categorization outcomes.

Table 4. The comparison of classification performance among various approaches

Works	Year	Samples (Malware/ Benign)	Technique	F-measure (%)	Classification Accuracy (%)
GIST+SVM	2011	4000/2000	ML	85,82	86,1
LBP+SVM	2018	4000/2000	ML	77,49	78,05
CLGM+SVM	2019	4000/2000	ML	91,98	92,06
DNN+SVM	2019	14 733/2486	DL	97,44	97,46
DCCNN-SMO+SVM	-	14 733/2486	DL with optimization methodologies	98,01	98,55
CB-STM-RENet based DCCNN-SMO-HPSO		14 733/2486	DL with optimization methodologies	98,65	98,90
CB-STM-RENet based DCCNN-EMRFO		14 733/2486	DL with optimization methodologies	99,01	99,4

Table 5. Comparison of Suggested Structure with the current CNN Models

Models	Accuracy (%)	F1-Score (%)	Precision (%)	MCC (%)	Recall (%)	AUC-PR (%)	AUC-ROC (%)
AlexNet	92,86	68,07	99,60	58,74	51,71	90,41	96,85
VGG 16	94,72	91,46	95,52	83,90	87,72	93,21	98,16
Inceptionv3	94,89	80,55	99,20	70,91	67,80	89,72	98,60
VGG19	95,38	83,53	99,02	74,29	72,23	90,88	97,39
Resnet50	95,62	82,82	99,71	73,79	70,82	94,32	98,48
IMDA	97,93	93,94	98,64	87,96	87,96	97,31	99,38
CB-STM-RENet	98,99	94,41	98,99	88,01	88,12	97,91	99,54
Ensemble model of CB-STM-RENet, Inception ResNet V2, and DenseNet 201	99,4	95,21	99,01	89,02	88,98	98,21	99,63

The efficacy of the ensemble model for IoT malware detection is also evaluated in comparison to other algorithms that are currently in use, including as AlexNet, VGG16, inceptionv3, VGG19, Resnet50, IMDA, and CB-STM-RENet. Better results are displayed in table 4. The suggested Ensemble models of CB-STM-RENet, Inception ResNet V2, and DenseNet 201, which employ region and boundary information systematically through the Avg and Max-pooling processes, demonstrate more thoroughly examined textural variety in the malware images. The features at various granularities were extracted with the aid of the channel boosting technique. The suggested structure outperformed the current approaches in terms effectiveness when the previously discussed CNN ideas were incorporated. This study used MCC, F1-Score, AUC-ROC, Accuracy, Precision, and Recall to quantify the impact for efficiency utilizing DL framework.

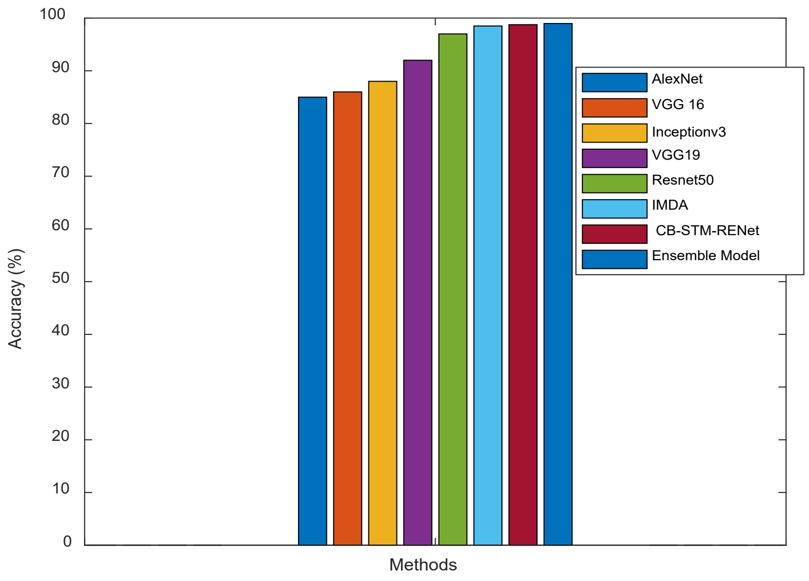


Figure 8. Accuracy Comparison

From the figure 8, it can be said that the suggested Ensemble method offers better accuracy value of 98,99 % which is explored textural variation in the malware images through systematically employing region and boundary information through the Avg and Max-pooling operations. The Suggested Ensemble method provides better accuracy value of 99,63 % when related with current models like AlexNet, VGG16, inceptionv3, VGG19, Resnet50, IMDA and CB-STM-RENet.

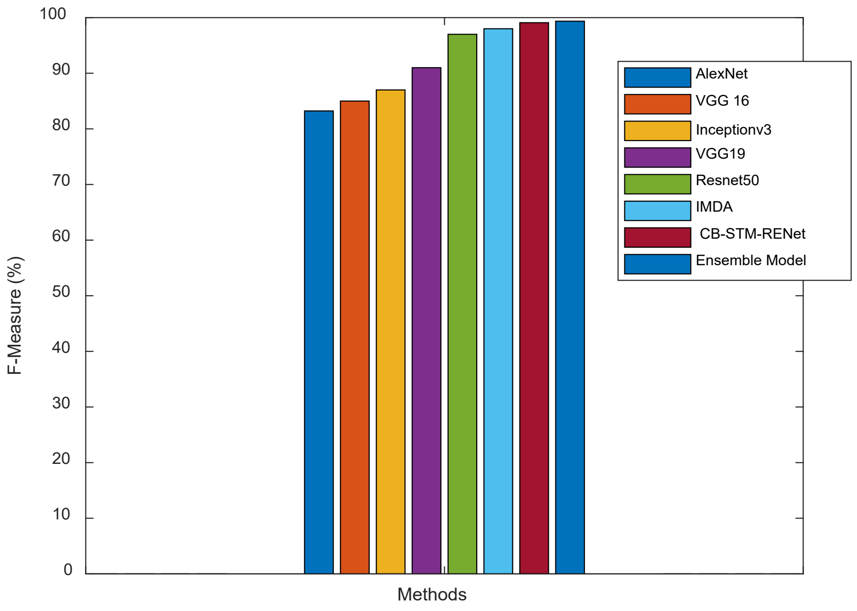


Figure 9. F1 Score Comparison

From the figure 9, it can be said that the suggested Ensemble method provides better F1 score value which is explored textural variation in the malware images by systematically using region and boundary information through the Avg and Max-pooling operations. The Proposed Ensemble method provides better F1-score value of 95,21 % when compared with current models like AlexNet, VGG16, inceptionv3, VGG19, Resnet50, IMDA and

CB-STM-RENet.

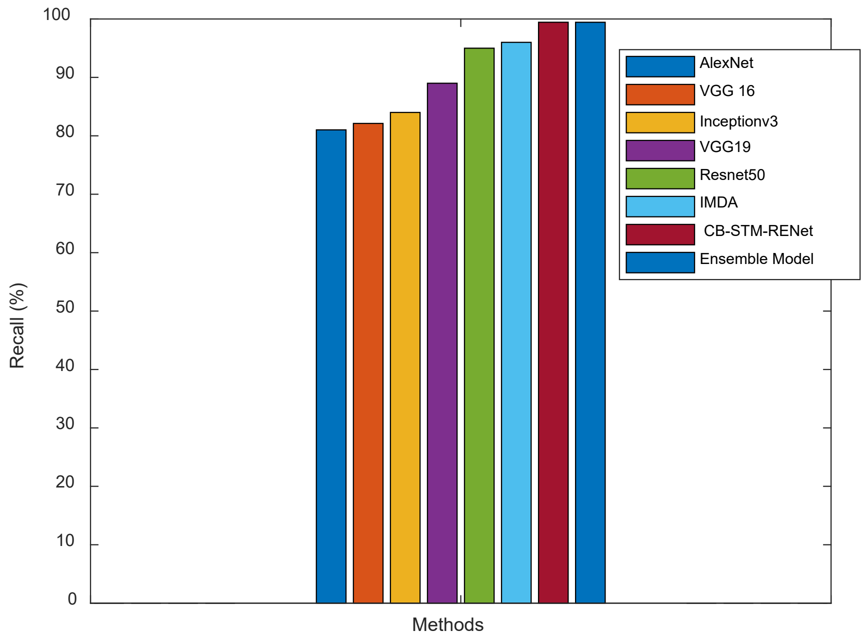


Figure 10. Recall Score Comparison

From the figure 10, it can be revealed that the suggested Ensemble method provides better recall value comparison. The Suggested Ensemble method provides better F1-score value of 99,01 % when compared with current models like AlexNet, VGG16, inceptionv3, VGG19, Resnet50, IMDA and CB-STM-RENet.

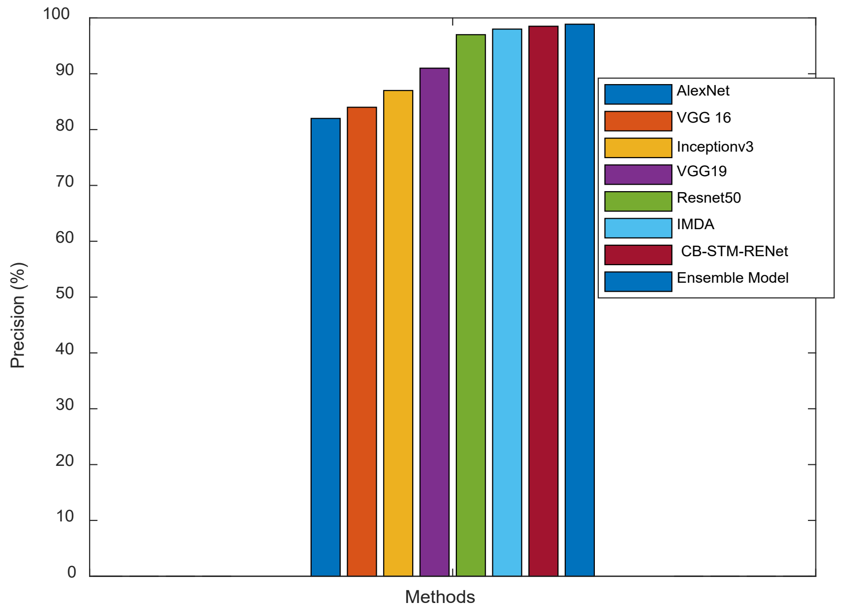


Figure 11. Precision Score Comparison

From the figure 11, it can be said that the suggested Ensemble method provides better precision value comparison. The Suggested Ensemble method provides better precision value of 95,21 % when compared with current models like AlexNet, VGG16, inceptionv3, VGG19, Resnet50, IMDA and CB-STM-RENet.

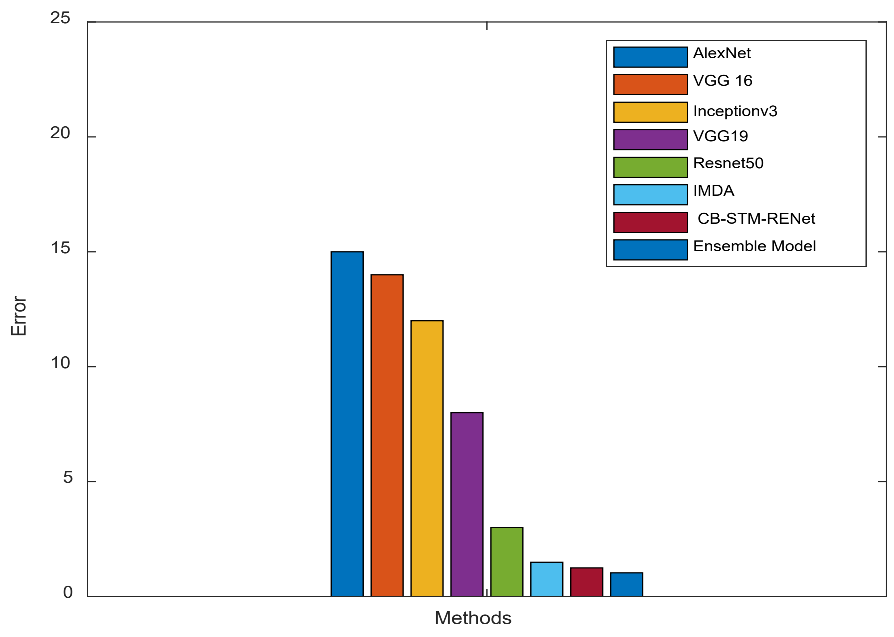


Figure 12. Error rate Comparison

From the figure 12, it can be said that the suggested Ensemble method obtained lesser error rate value comparison. The Suggested Ensemble method provides better error rate value when compared with current frameworks like AlexNet, VGG16, inceptionv3, VGG19, Resnet50, IMDA and CB-STM-RENet.

CONCLUSION

The identification of malware threats for securing IoT-based cyber spaces is addressed in this study. A combinatorial optimisation procedure termed CB-STM-RENet-based DCCNN-EMRFO for detecting and identifying malware is suggested in this work. Preprocessing is initially performed to transform malware files into picture files, which improves malware visualisation. These malware features were later incorporated into CB-STM-RENet Ensemble, Inception ResNet V2, and DenseNet 201. Experiments reveal that the combined strategy outperforms other methods currently in use in terms of classification outcomes. The suggested approach delves deeper into the variety of textures in malware photos by systematically leveraging area and border information using Avg and Max clustering algorithms. The channel enhancement approach was used to extract features with varying degrees of precision. When the previously described CNN principles were combined with the suggested architecture, it outperformed existing models. This study uses the DL architecture to measure performance and reflect its significance based on obtained values of MCC, F1 scores, AUC-ROC, precision, recall and accuracies.

REFERENCES

1. Schatz D, Bashroush R, and Wall J. Towards a more representative definition of cyber security. Journal of Digital Forensics, Security and Law, 12(2), pp. 53-73. https://doi.org/10.15394/jdfsl.2017.1476.

2. Dev H, Sen T, Basak M, and Ali ME. An approach to protect the privacy of cloud data from data mining based attacks. In SC Companion: High Performance Computing, Networking Storage and Analysis, pp. 1106-1115. https://doi.org/10.1109/SC.Companion.2012.133.

3. Li Y, Gai K, Qiu L, Qiu M, and Zhao H. Intelligent cryptography approach for secure distributed big data storage in cloud computing. Information Sciences, 387, pp. 103-115. https://doi.org/10.1016/j.ins.2016.09.005.

4. Kumar CO, Tejaswi K, and Bhargavi P. A distributed cloud-prevents attacks and preserves user privacy. In 15th International Conference on Advanced Computing Technologies (ICACT), pp. 1-6. https://doi.org/10.1109/ICACT.2013.6710509.

5. Om Kumar CU, and Sathia Bhama PR. Detecting and confronting flash attacks from IoT botnets. The Journal of Supercomputing, 75, pp. 8312-8338. https://doi.org/10.1007/s11227-019-03005-2.

6. Singh NK, Kumar CO, and Sridhar R. Flash crowd prediction in Twitter. In 4th International Conference on Advanced Computing and Communication Systems (ICACCS), pp. 1-6. https://doi.org/10.1109/ICACCS.2017.8014676.

7. CU OK, and Sathia Bhama PR. Efficient ensemble to combat flash attacks. Computational Intelligence, 40(1), pp. e12488. https://doi.org/10.1111/coin.12488.

8. Om Kumar CU, Durairaj J, Ahamed Ali SA, Justindhas Y, and Marappan S. Effective intrusion detection system for IoT using optimized capsule auto encoder model. Concurrency and Computation: Practice and Experience, 34(13), pp. e6918. https://doi.org/10.1002/cpe.6918

9. Om Kumar CU, and Sathia Bhama PR. Proficient detection of flash attacks using a predictive strategy. In Emerging Research in Computing, Information, Communication and Applications: ERCICA, 1, pp. 367-379. https://doi.org/10.1007/978-981-16-1338-8_32.

10. Om Kumar CU, Marappan S, Murugeshan B, and Beaulah PMR. Intrusion detection model for IoT using recurrent kernel convolutional neural network. Wireless Personal Communications, 129(2), pp. 783-812. https://doi.org/10.1007/s11277-022-10155-9.

11. Rawat R, Gupta S, Sivaranjani S, Cu OK, Kuliha M, and Sankaran KS. Malevolent information crawling mechanism for forming structured illegal organisations in hidden networks. International Journal of Cyber Warfare and Terrorism (IJCWT), 12(1), pp. 1-14. https://www.igi-global.com/article/malevolent-information-crawling-mechanism-for-forming-structured-illegal-organisations-in-hidden-networks/311422#:~:text=DOI%3A%20 10.4018/IJCWT.311422.

12. Kumar CO, Bhama PRS, and Prasad. Efficacious intrusion detection on cloud using improved BES and HYBRID SKINET-EKNN. In Emerging Research in Computing, Information, Communication and Applications: Proceedings of ERCICA, pp. 61-72. https://doi.org/10.1007/978-981-19-5482-5_6.

13. CU OK, Pranavi D, Laxmi BA, and Devasena R. Variational autoencoder for IoT botnet detection. In Using Computational Intelligence for the Dark Web and Illicit Behavior Detection, pp. 74-88. https://www.igi-global.com/chapter/variational-autoencoder-for-iot-botnet-detection/307871#:~:text=DOI%3A%20 10.4018/978%2D1%2D6684%2D6444%2D1.ch005.

14. Wikipedia Malware. [(accessed on 6 December 2022)]. Available online: https://en.wikipedia.org/wiki/Malware.

15. Financesonline.com Number of Smartphone and Mobile Phone Users Worldwide in 2022/2023: Demographics, Statistics, Predictions. [(accessed on 11 December 2022)]. Available online: https://financesonline.com/number-of-smartphone-users-worldwide/

16. Lee H, Park J, and Lee U. A systematic survey on android api usage for data-driven analytics with smartphones. ACM Computing Surveys, 55(5), pp. 1-38. https://doi.org/10.1145/3530814.

17. Mercaldo F, Nardone V, Santone A, and Visaggio CA. Ransomware steals your phone. formal methods rescue it. In Formal Techniques for Distributed Objects, Components, and Systems: 36th IFIP WG 6.1 International Conference, FORTE, Held as Part of the 11th International Federated Conference on Distributed Computing Techniques, DisCoTec Proceedings 36, pp. 212-221. https://doi.org/10.1007/978-3-319-39570-8_14.

18. Marulli F, and Visaggio CA. Adversarial deep learning for energy management in buildings. In Proceedings of the Summer Simulation Conference, pp. 1-11.

19. Campanile L, Iacono M, Levis AH, Marulli F, and Mastroianni M. Privacy regulations, smart roads, blockchain, and liability insurance: Putting technologies to work. IEEE Security & Privacy, 19(1), pp. 34-43. https://doi.org/10.1109/MSEC.2020.3012059.

20. Malware Statistics in 2023: Frequency, Impact, Cost & More. [(accessed on 10 December 2022)]. Available online: https://www.comparitech.com/antivirus/malware-statistics-facts/

21. April 12, 2021—Check Point Software. [(accessed on 11 December 2022)]. Available online: https://blog.checkpoint.com/2021/04/12/

22. Google Safe Browsing—Google Transparency Report. [(accessed on 7 December 2022)]. Available online: https://transparencyreport.google.com/safe-browsing/overview?hl=en_GB&unsafe=dataset:1;series:malwareDetected,phishingDetected;start:1148194800000;end:1612080000000&lu=unsafe

23. Statista Our Research and Content Philosophy. [(accessed on 7 December 2022)]. Available online: https://www.statista.com/aboutus/our-research-commitment

24. Global Ransomware Damage Costs Predicted to Exceed $265 Billion By 2031. [(accessed on 7 December 2022)]. Available online: https://cybersecurityventures.com/global-ransomware-damage-costs-predicted-to-reach-250-billion-usd-by-2031/#:~:text=2022%20Ransomware%20Market%20Report%20is%20sponsored%20by%20KnowBe4&text=The%20damages%20for%202018%20were,than%20it%20was%20in%202015.

25. Khan RU, Zhang X, and Kumar R. Analysis of ResNet and GoogleNet models for malware detection. Journal of Computer Virology and Hacking Techniques, 15, pp. 29-37. https://doi.org/10.1007/s11416-018-0324-z.

26. Muzaffar A, Hassen HR, Lones MA, and Zantout H. An in-depth review of machine learning based Android malware detection. Computers & Security, 121, pp. 102833. https://doi.org/10.1016/j.cose.2022.102833

27. Deng H, Guo C, Shen G, Cui Y, and Ping Y. MCTVD: A malware classification method based on three-channel visualization and deep learning. Computers & Security, 126, p.103084. https://doi.org/10.1016/j.cose.2022.103084.

28. Su J, Vasconcellos DV, Prasad S, Sgandurra D, Feng Y, and Sakurai K. Lightweight classification of IoT malware based on image recognition. In IEEE 42Nd annual computer software and applications conference (COMPSAC), 2, pp. 664-669. https://doi.org/10.1109/COMPSAC.2018.10315.

29. Ren Z, Wu H, Ning Q, Hussain I, and Chen B. End-to-end malware detection for android IoT devices using deep learning. Ad Hoc Networks, 101, pp.102098. https://doi.org/10.1016/j.adhoc.2020.102098.

30. Hussain SJ, Ahmed U, Liaquat H, Mir S, Jhanjhi NZ, and Humayun M. IMIAD: intelligent malware identification for android platform. In International Conference on Computer and Information Sciences (ICCIS), pp. 1-6. https://doi.org/10.1109/ICCISci.2019.8716471.

31. Shafiq M, Tian Z, Bashir AK, Du X, and Guizani M. CorrAUC: a malicious bot-IoT traffic detection method in IoT network using machine-learning techniques. IEEE Internet of Things Journal, 8(5), pp. 3242-3254. https://doi.org/10.1109/JIOT.2020.3002255.

32. Zhang Y, Yang Y, and Wang X. A novel android malware detection approach based on convolutional neural network. In Proceedings of the 2nd international conference on cryptography, security and privacy, pp. 144-149.

33. Xu K, Li Y, Deng RH, and Chen K. Deeprefiner: Multi-layer android malware detection system applying deep neural networks. In IEEE European Symposium on Security and Privacy (EuroS&P), pp. 473-487. https://doi.org/10.1109/EuroSP.2018.00040.

34. Alzaylaee MK, Yerima SY, and Sezer S. DL-Droid: Deep learning based android malware detection using real devices. Computers & Security, 89, pp. 101663. https://doi.org/10.1016/j.cose.2019.101663.

35. Bendiab G, Shiaeles S, Alruban A, and Kolokotronis N. IoT malware network traffic classification using visual representation and deep learning. In 6th IEEE Conference on Network Softwarization (NetSoft), pp. 444-449. https://doi.org/10.1109/NetSoft48620.2020.9165381.

36. Parra GDLT, Rad P, Choo KKR, and Beebe N. Detecting Internet of Things attacks using distributed deep learning. Journal of Network and Computer Applications, 163, pp. 102662. https://doi.org/10.1016/j.jnca.2020.102662.

37. HaddadPajouh H, Dehghantanha A, Khayami R, and Choo KKR. A deep recurrent neural network based approach for internet of things malware threat hunting. Future Generation Computer Systems, 85, pp.88-96. https://doi.org/10.1016/j.future.2018.03.007.

38. Rehman MU, Shafique A, Khalid S, Driss M, and Rubaiee S. Future forecasting of COVID-19: a supervised learning approach. Sensors, 21(10), pp. 1-17. https://doi.org/10.3390/s21103322.

39. Driss M, Almomani I, e Huma Z, and Ahmad J. A federated learning framework for cyberattack detection in vehicular sensor networks. Complex & Intelligent Systems, 8(5), pp.4221-4235. https://doi.org/10.1007/s40747-022-00705-w.

40. Bozkir AS, Cankaya AO, and Aydos M. Utilization and comparision of convolutional neural networks in malware recognition. In 27th signal processing and communications applications conference (SIU), pp. 1-4.

41. Szegedy C, Ioffe S, Vanhoucke V, and Alemi, A. Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the AAAI conference on artificial intelligence, 31(1), pp. 4278- 4284. https://doi.org/10.1609/aaai.v31i1.11231.

42. Huang G, Liu Z, Van Der Maaten L, and Weinberger KQ. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4700-4708. https://doi.org/10.48550/arXiv.1608.06993.

43. Elmasry, "IOT_Malware." https://www.kaggle.com/anaselmasry/iot-malware (accessed ` Aug. 08, 2021)

44. Bandara U, and Wijayrathna G. Detection of source code plagiarism using machine learning approach. Int J Comput Theory Eng, 4(5), pp. 674-678.

FINANCING
"The authors did not receive financing for the development of this research".

CONFLICT OF INTEREST
"The authors declare that there is no conflict of interest".

AUTHORSHIP CONTRIBUTION
Conceptualization: P.Vijayalakshmi.
Data curation: D. Karthika.
Formal analysis: D. Karthika.
Research: P.Vijayalakshmi.
Methodology: D. Karthika.
Drafting - original draft: P.Vijayalakshmi.