Journal of Theoretical and Applied Information Technology
30th September 2024. Vol.102. No. 18
© Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ENHANCING IOT SECURITY THROUGH MALWARE CLASSIFICATION WITH HYBRID DEEP LEARNING MODEL GENERATIVE ADVERSARIAL NETWORK AND DEEP BELIEF NETWORK (HCAGAN_DBN)

V.S. JEYALAKSHMI[1], KRISHNAN NALLAPERUMAL[2]

[1]Research Scholar, Centre for Information Technology and Engineering, Manonmaniam Sundaranar University, Abishekapatti, Tirunelveli-627012, Tamil Nadu, India

[2]Senior Professor (Retd), Centre for Information Technology and Engineering, Manonmaniam Sundaranar University, Abishekapatti, Tirunelveli-627012, Tamil Nadu, India

E-mail: [1]vsjeyalakshmiap@gmail.com, [2]krishnan17563@gmail.com

ID 55502 Submission	Editorial Screening	Conditional Acceptance	Final Revision Acceptance
04-09-24	04-09-2024	22-09-2024	26-09-2024

ABSTRACT

IoT has gained popularity as a result of the advancement and promise of smart technologies. The necessity of IoT technology has been accompanied by an increase in security issues about IoT devices, apps and infrastructure. Due to the wide range of abilities of IoT devices, dynamic and the constantly changing environment, enhanced system security measures are challenging and it is risky to only implement the most fundamental security requirements. The computer system is significantly at risk from malicious software (Malware). Finding malicious intent in a program is a crucial responsibility for security purpose. Here, novel method is proposed to detect significantly the previously unidentified threats in a cyber security land space. A new hybrid model HCAGAN-DBN is developed to classify the malware family efficiently. GAN architecture has generator and discriminator, the re-sampled output data is transformed into DBN for malware family classification in a zero short learning. Generative Adversarial Network and Deep Belief Network based intrusion detection model is proposed in this paper for Malware classification in IoT environment. The proposed model trained for one-dimensional images which learn and analyzes the characteristics of the complicated patterns of the Byte files and the Assembly files. Experiments were carried out with the Microsoft malware Challenge Dataset (MMCD). The outcomes of the evaluation show that with an average accuracy of 99.83% our HCAGAN_DBN Classifier performs better than traditional state-of-the-art works.

Keywords: *Internet of Things, Generative Adversarial Network, Cyber Security, Malware Analysis, Deep Belief Network*

1. INTRODUCTION

The Internet of Things (IoT) is the largest digital megatrend that bridges the digital and physical realms. As people, objects, technology and the Internet become more interconnected, new business models and ways for individuals to interact with other living things around the world are also emerging [1]. IoT devices are frequently target by cybercriminals who infiltrate them using malware, obsolete firmware, weak authentication and other techniques. This is due to their sophisticated hardware and software design, implementations, as well as their lack of security features and protections.

Due to the industry's swift adoption of IoT technologies, these threats will keep rising. Malware is one of the main risks to IoT devices. In October 2016, the Mirai virus family attacked Dyn, a prominent US DNS service provider, with one of the largest and most effective DDoS attacks [2]. The infection, which also targeted other well-known internet companies like Google, Amazon, etc., are infected over 1.2 million IoT devices.

Malware can be analyzed using either a dynamic or static approach. With the dynamic technique, running files are observed in a regulated setting such as an emulator to capture their behavior [3]. This approach necessitates processing infected data in a covert manner. Anti-virtual machine and anti-emulator techniques guard against some malware types; the virus is altered to function normally when these conditions are identified. This alteration shields us from their nefarious actions [41].

The static approach of malware analysis reads raw malware bytes to determine the behavioral (conceived and disclosed) properties of the raw files, frequently with the aid of some tools and subject-matter knowledge. The analyst can examine all the potential malware execution possibilities that were not visible during the attack by carefully examining the static code. It is the greatest way for classifying malware since it gives analysts a wealth of knowledge about malware [5, 6]. Malware must be categorized into several classes in order to be detected, as malware detection algorithms heavily rely on the traits that each class of malware represents. Families of malware behave differently and affect a computer system in different ways.

To increase the detection rate, these procedures make use of a variety of technologies and techniques, including heuristics, machine learning and data analytics. There are several approaches to malware detection that make use of the tools and procedures indicated above. Using a signature-based approach is an effective way to combat known and related malware variants. It is unable to locate malware that has not yet been discovered, though. Behavior-based, heuristic-based, and model-checking-based malware detection systems are effective in locating unknown malware components, but they are not very good at locating complex malware versions that use packaging and obfuscation techniques. For financial benefit, cyber criminals use obfuscated techniques to create new malware. Here, a novel hybrid model HCAGAN-DBN is developed to discover the previously unseen malware with a promising result.

Deep learning-based approaches are being used as a new paradigm in malware detection and classification to address the shortcomings of existing approaches. But its application in the field of cyber security has been lacking, especially when it comes to virus detection. The "deep learning" subgroup of artificial intelligence is built on ANNs. Deep learning (DL) uses multiple hidden layers and learns from examples. To enhance model performance, a range of DL methods have been used recently, such as convolutional neural networks (CNN), recurrent neural networks (RNN), deep belief networks (DBN) and deep neural networks (DNN). There are many advantages to deep learning over traditional learning paradigm.

This paper suggests a unique architecture for malware categorization based on hybrid deep learning. The present HCAGAN_DBN analyzes both byte and asm feature and provides the efficient result in detecting the malware families. The byte feature is process based on 1D vector instead of 2D feature. Since malicious files might have different dimensions, a large portion of the original data will be lost if the virus picture has fixed width and length. In order to preserve as many of the malware's original properties as possible, it is turned into a square picture with a length equal to the file size squared. The malware image is scaled to a 128 by 128 matrix so that the texture of the image is retained during neural network training. The process of converting images has some issues. Therefore, the succeeding classification function will classify this virus as belonging to a separate family. According to the aforementioned study and conjecture, the detection outcome may not be optimum if the 2D feature of the malware image is employed directly for classification. The Microsoft malware dataset is used for the evaluation.

2. RELATED WORKS

Yuan et al [7] proposed MDMC approach for byte-level malware classification uses deep learning and markov images. Malware binaries are transformed into markov images using bytes transfer probability matrices as the core component of MDMC. Then, for the classification of markov images, a deep convolutional neural network is employed. Two malware datasets, the Microsoft dataset and the Drebin dataset, are used in the tests.

Gibert et al. [8] introduce a unique malware classification system with a modular design that blends hand-engineered features and end-to-end components. The multimodal technique learns and combines malware features from many information sources, leading to better classification performance than classifiers that only accept one type of data as input. The Microsoft Malware Classification benchmark has been used to assess how well our multimodal learning system performs.

For malware categorization, ömer Aslan and Abdullah Asim Yilmaz [9] suggest a revolutionary hybrid deep-learning based architecture. The DL system receives grayscale photos of the malware samples first. The proposed method uses the convolution layers of the suggested hybrid architecture to extract high-level malware features from malware images after the image acquisition portion is complete. Finally, the system receives supervised training. In the recommended model, a hybrid model is created by combining a number of comprehensive deep-learning models that rely on a transfer-learning approach. A number of hidden layers and the Rectified Linear Unit (ReLU) function are used during the aforementioned operations.

Darem et al. [10] provide an adaptive learning model to account for the novel malware variants. Sequential deep learning and concept drift detection are employed in this model. By running the programs in a sandbox environment and gathering their Application Programming Interface (API) traces, virus behaviors were retrieved through the use of dynamic analysis. To find the traits that varied between malicious variations, the malware samples were categorized according to how they first manifested.

In addition, a powerful malware detection method is proposed utilizing a deep LSTM model by Aiyshwariya Devi and Arunachalam [11] for improved IoT device security in edge nodes. Contextual features are used in the initial process to differentiate between attack and normal nodes using a trust value. Following the discovery of attack nodes, these are taken into account for forecasting the various attacks that might be present in the network, with some preprocessing and feature extraction techniques being used for efficient classification. This malware detection method makes use of the Deep LSTM classifier. After completing malware detection, the Improved Elliptic Curve Cryptography (IECC) technique is used to perform prevention.

Al-Andoli et al. [12] introduced an ensemble-based Deep Learning classifier for malware detection. Specifically, a stacked ensemble learning method is developed, employing five Deep Learning basis models and a neural network as a Meta model. To train and enhance the DL models, a hybrid optimization method based on the BP and Particle Swarm Optimization (PSO) algorithms is employed. A parallel computing architecture is used to increase the ensemble method's scalability and effectiveness.

Table 1: Survey on Malware Detection-2023

Author	Year	Dataset	Model	Accuracy
Bhavya Dawra et al.	2023	Malimg	Transfer Learning	98.04%
Kwok Taichui	2023	Malimg	GAN	95.0%
Osho Sharma et al.	2023	Malimg	GAN-Transfer Learning	99.5%

Table 1 depicts the survey of malware categorization on 2023. From the literature of malware classification in 2023, all the authors are using deep learning algorithms like Transfer learning, GAN, etc., to classify the malware family, but the time complexity is not achieved. The proposed hybrid model HCAGAN-DBN achieves the time complexity and reached a good accuracy of 99.83%.

3. PROPOSED METHODOLOGY

In this study, HCAGAN_DBN, a novel framework is introduced for the detection and classification of malware that combines several types of characteristics to identify correlations across multiple modalities. Our methodology draws knowledge from a variety of sources to optimize the advantages of several feature types and represent the properties of malware executable. To properly describe malware features, we suggest a basic system made up of end-to-end components that combines the advantages of feature engineering and deep learning. Figure 1 depicts the proposed malware classification framework.

3.1 Microsoft Malware Challenge Dataset (MMCD)

For the Big Data Innovators Gathering Challenge from 2015, Microsoft contributed over half a terabyte of malicious malware (Ronen et al., 2018). The dataset is currently publicly available and hosted on Kaggle1. The dataset has evolved into the de facto standard for assessing machine learning methods for the task of classifying malware. Each sample in the set is identifiable by a hash and its class, an integer corresponding to one of the 9 malware families. The set of samples comprises 9 different malware families.

The Dataset contains hexadecimal and assembly language source code. The machine code is displayed in hexadecimal form in the hex view. Each line is made up of a collection of consecutive 16-byte integers and the memory's starting address for the machine codes. One can calculate an executable's structural entropy, represent its binary content as a grayscale image, extract byte n-grams, and more using this kind of representation.

The symbolic machine code of the executable is contained in the assembly language source code, together with metadata such basic function calls, memory allocation, and variable information.

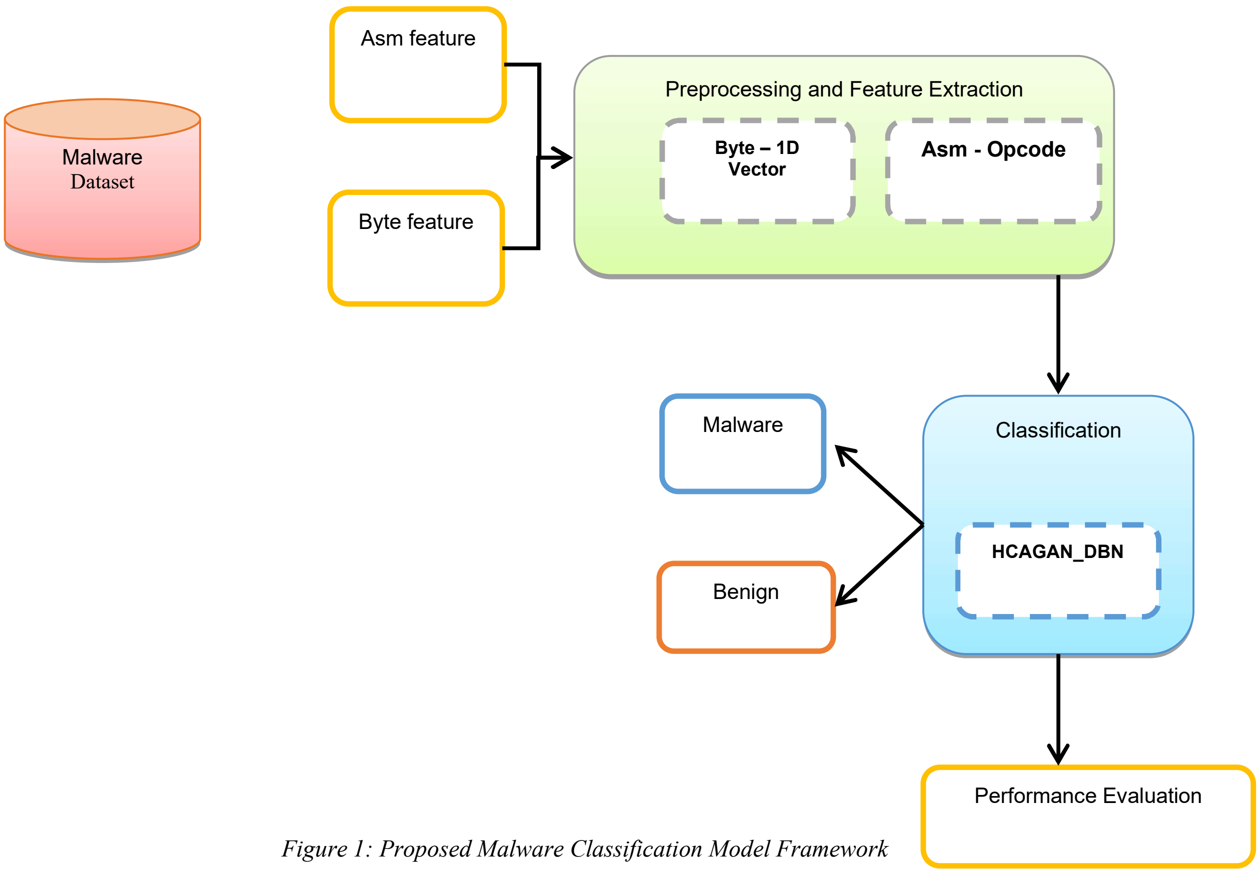
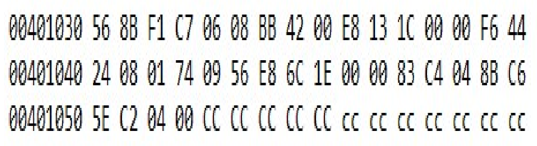
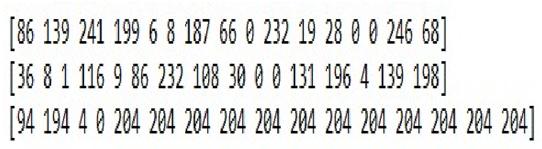


Figure 1: Proposed Malware Classification Model Framework



```
00401030 56 8B F1 C7 06 08 BB 42 00 E8 13 1C 00 00 F6 44
00401040 24 08 01 74 09 56 E8 6C 1E 00 00 83 C4 04 8B C6
00401050 5E C2 04 00 CC CC CC CC CC cc cc cc cc cc cc cc
```

Figure 2: Hexa decimal byte samples



```
[86 139 241 199 6 8 187 66 0 232 19 28 0 0 246 68]
[36 8 1 116 9 86 232 108 30 0 0 131 196 4 139 198]
[94 194 4 0 204 204 204 204 204 204 204 204 204 204 204 204]
```

Figure 3: Conversion of hexadecimal bytes into 1D vector

3.1.1 1D conversion of byte feature

The malware's binary content is extracted from the file header opening and transformed into a decimal one-dimensional vector in order to identify the malware family. The initial 1024 bytes of the infection are transformed by Algorithm 1 into a one-dimensional vector that serves as the malware sample's malhash byte feature.

The bytes are extracted from each file and converted them into the integer format and stored in vector then it is converted to ID vector through flatten function. The steps involved in converting the bytes to 1D vector is given in algorithm 1.

Figure 2 depicts a hexadecimal malware sample, and Figure 3 depicts the transformation of this sample into a decimal 1D vector, for instance, '56 8D' is transformed into '86 141'. A security analyst cannot retrieve the privacy content of the original file in reverse because the 1D conversion technique only extracts a portion of the non-semantic information from the malware header. Therefore, by disclosing the tenants' data, this technique will not breach their privacy.

Algorithm 1: Generating 1D vector from bytes

```
Input: mal_file
Output: ID vector
pixelnumer=1024
f = open(mal_file, 'r')
c = f.readlines()
pixelnumber = []
w = 16
for L in content:
L = L[8:]
for i in range(0, len(L), 2):
p = [int(L[i:i + 2],16)] pixelvecr.append(p)
if len(pixelvec) > pixelnumber :
break
onedim = numpy.array(pixelvec).flatten()
return onedim[0: pixelnumber]
```

Algorithm 1: Generating 1D vector from bytes

```
Input: mal_file
Output: ID vector
pixelnumer=1024
f = open(mal_file, 'r')
c = f.readlines()
pixelnumber = []
w = 16
for L in content:
L = L[8:]
for i in range(0, len(L), 2):
p = [int(L[i:i + 2],16)] pixelvecr.append(p)
if len(pixelvec) > pixelnumber :
break
onedim = numpy.array(pixelvec).flatten()
return onedim[0: pixelnumber]
```

3.1.2 Opcode extraction

The Microsoft PE program format [14] consists of sections, section headers, a PE header, a DOS header and other data. These parts include the import and export capabilities, resource details and executable scripts that are crucial indicators of the infection. By studying its disassembled codes, malware's workflow can be easily comprehended.

First, the essential features of the deconstructed virus are extracted, including file size, opcode line count and information entropy. Figure 4 displays a sample of the 0AguvpOCcaf2myVDYFGb.asm file's dissected code segment. In this example, the address information, comments, opcodes[17], registers and parameters are all included in the assembly code.

Table 2 displays the characteristics were taken from assembly code. The study of samples revealed that different types of assembly files have varying file formats. It is necessary to locate the actual code section contents because certain disassembled executable code sections do not begin with ".text".

3.2 Classification Model- HCAGAN_DBN

The deep belief network (DBN), which was first put forth by Hinton [15], is one of the most well-known deep learning techniques. This algorithm picks up new information quickly and can find the perfect parameters faster than the others. The essential elements of a conventional DBN are a restricted Boltzmann machine (RBM)-based unsupervised learning module and a logistic regression layer [16].

The Restricted Boltzmann Machine (RBM), a well-known stochastic neural network, builds a deep belief network (DBN) by layer-wise training. A layer of hidden Boolean neurons and a layer of binary-valued neurons are both present in the RBM. Despite being between the layers, the connections between the neurons in the same layer are not symmetric or bidirectional.

A layer-wise configuration learns a probability distribution between the two levels based on the energy function of the configuration, which is specified in Eq. (1). The probability distribution is thus expressed by the following equation (2).

$$Ef(a,b) = -\sum_{m=1}^{z_a} x_m a_m - \sum_{n=1}^{z_b} y_n b_n - \sum_{m=1}^{z_a}\sum_{n=1}^{z_b} b_n W_{n,m} a_m \quad (1)$$

$$pd = \frac{e^{-Ef(a,b)}}{\sum_c \sum_b e^{-Ef(c,b)}} \quad (2)$$

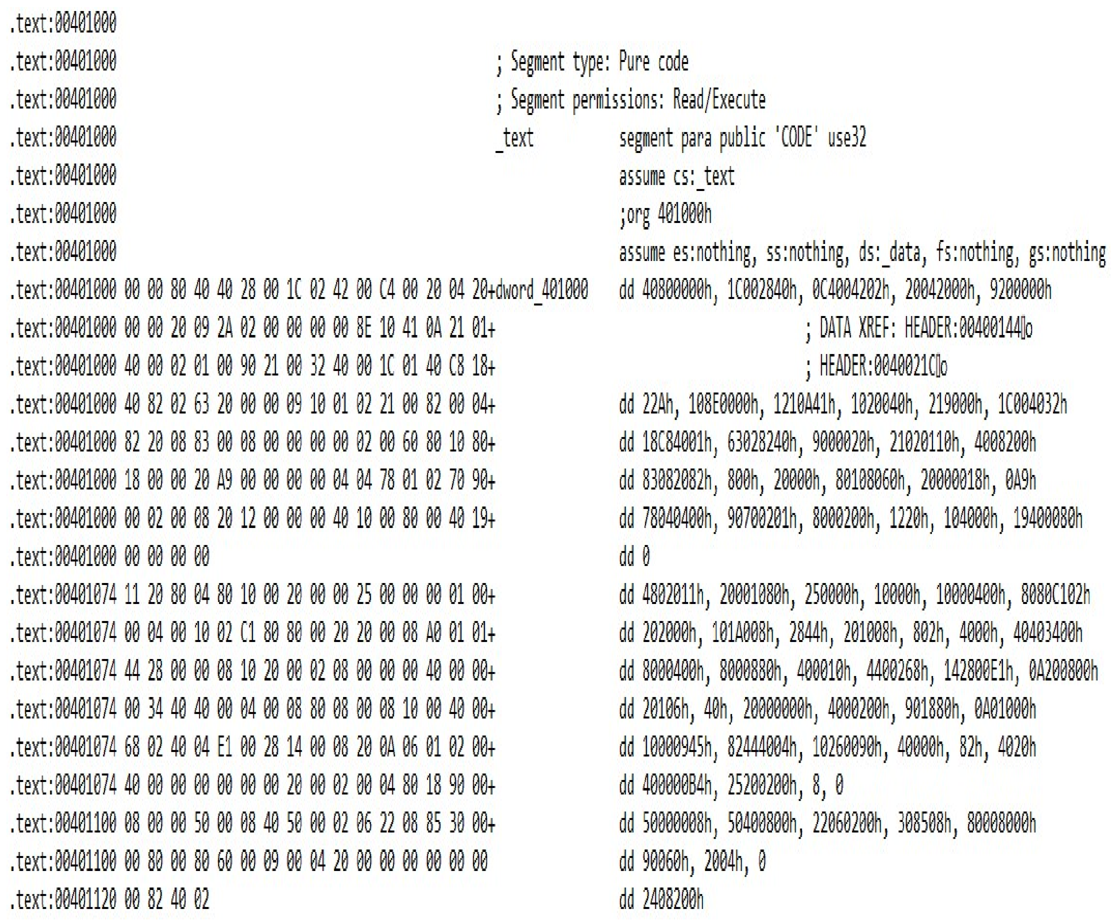
The weight matrices that separate the hidden layer from the visible layer are a_m and b_n, the biases for the two layers are x_m and y_n, and the number of neurons in the visible layer is $W_{n,m}$, the number of Boolean hidden neurons in the hidden layer is b_n.

The activation probability functions are then represented in an equation (3) and (4).

$$pd(a_m = 1|b) = sig\left(x_m + \sum_{n=1}^{i_b} W_{n,m} b_n\right) \quad (3)$$

$$pd(b_n = 1|a) = sig\left(y_n + \sum_{m=1}^{i_a} W_{n,m} a_m\right) \quad (4)$$

Additionally, sig() is used to represent the logistic sigmoid function. Since the weight matrices and layer biases can be trained unsupervised, the pre-training principles support this. A single hidden restricted Boltzmann machine (RBM) cannot capture the peculiarities of the data. A deep belief network (DBN), which is created by stacking layers of RMBs in a hierarchical fashion and concluding with a logistic regression layer, may gradually extract deep characteristics from the input dataset. The first RBM of the DBN is pre-trained as an independent RBM using the training data as inputs.



```
.text:00401000
.text:00401000                                           ; Segment type: Pure code
.text:00401000                                           ; Segment permissions: Read/Execute
.text:00401000                                           _text          segment para public 'CODE' use32
.text:00401000                                                          assume cs:_text
.text:00401000                                                          ;org 401000h
.text:00401000                                                          assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
.text:00401000 00 00 80 40 40 28 00 1C 02 42 00 C4 00 20 04 20+dword_401000   dd 40800000h, 1C002840h, 0C4004202h, 20042000h, 9200000h
.text:00401000 00 00 20 09 2A 02 00 00 00 00 8E 10 41 0A 21 01+                                 ; DATA XREF: HEADER:00400144o
.text:00401000 40 00 02 01 00 90 21 00 32 40 00 1C 01 40 C8 18+                                 ; HEADER:0040021Co
.text:00401000 40 82 02 63 20 00 00 09 10 01 02 21 00 82 00 04+             dd 22Ah, 108E0000h, 1210A41h, 1020040h, 219000h, 1C004032h
.text:00401000 82 20 08 83 00 08 00 00 00 00 02 00 60 80 10 80+             dd 18C84001h, 63028240h, 9000020h, 21020110h, 4008200h
.text:00401000 18 00 00 20 A9 00 00 00 00 04 04 78 01 02 70 90+             dd 83082082h, 800h, 20000h, 80108060h, 20000018h, 0A9h
.text:00401000 00 02 00 08 20 12 00 00 00 40 10 00 80 00 40 19+             dd 78040400h, 90700201h, 8000200h, 1220h, 104000h, 19400080h
.text:00401000 00 00 00 00                                                  dd 0
.text:00401074 11 20 80 04 80 10 00 20 00 00 25 00 00 00 01 00+             dd 4802011h, 20001080h, 250000h, 10000h, 10000400h, 8080C102h
.text:00401074 00 04 00 10 02 C1 80 80 00 20 20 00 08 A0 01 01+             dd 202000h, 101A008h, 2844h, 201008h, 802h, 4000h, 40403400h
.text:00401074 44 28 00 00 08 10 20 00 02 08 00 00 00 40 00 00+             dd 8000400h, 8000880h, 400010h, 4400268h, 142800E1h, 0A200800h
.text:00401074 00 34 40 40 00 04 00 08 80 08 00 08 10 00 40 00+             dd 20106h, 40h, 20000000h, 4000200h, 901880h, 0A01000h
.text:00401074 68 02 40 04 E1 00 28 14 00 08 20 0A 06 01 02 00+             dd 10000945h, 82444004h, 10260090h, 40000h, 82h, 4020h
.text:00401074 40 00 00 00 00 00 00 20 00 02 00 04 80 18 90 00+             dd 400000B4h, 25200200h, 8, 0
.text:00401100 08 00 00 50 00 08 40 50 00 02 06 22 08 85 30 00+             dd 50000008h, 50400800h, 22060200h, 308508h, 80008000h
.text:00401100 00 80 00 80 60 00 09 00 04 20 00 00 00 00 00 00              dd 90060h, 2004h, 0
.text:00401120 00 82 40 02                                                  dd 2408200h
```

Figure 4: Assembly source code sample (0AguvpOCcaf2myVDYFGb.asm)

Table 2: Extracted Features Description

Extracted Feature	Description
Basic	File entropy, line count and size
Section count	Line count for each file's top ten parts
Opcode count	Top 100 opcodes in every asm file
Opcode n_gram	In the disassembled files, there are a lot of common system call functions.
Stdcall count	Statistics on the quantity of frequently used system call operations in disassembled files

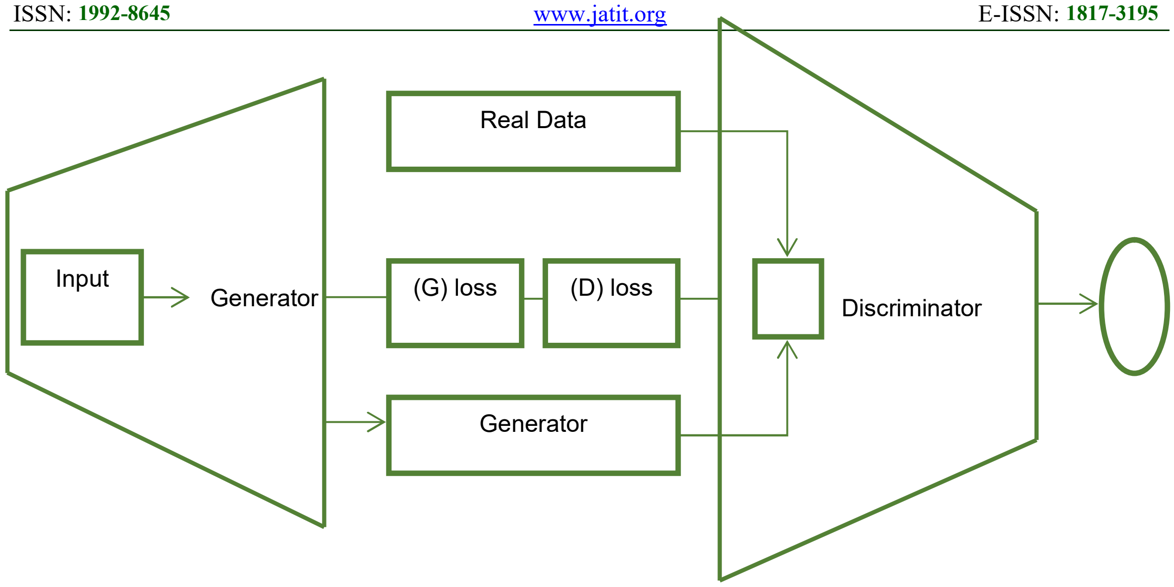


Figure 5: Architecture of GAN Network

Once the first RBM's weight matrix and bias settings are established, the output of the first RBM is selected to serve as the input for the second RBM. The hidden layers of the first two RBMs are then trained repeatedly using the same procedure, acting as a new RBM. The final phase entails layering an all-encompassing predictor (such a layer for logistic regression) on top of the network and training it under close supervision. After applying the aforementioned stages, the back-propagation (BP) approach is utilized for fine tuning to slightly change the parameters of the entire trained network.

GAN learns to produce new data from training sets that are similar to the training sets. Figure 5 depicts the GAN architecture. It has two crucial parts, a generator (G) and a discriminator (D). The discriminator is a model used to categorize instances as real (from the domain) or fake (created), whereas the generator is a model used to generate new convincing examples from the problem domain.

Using a generator network, a sample of data that resembles genuine data is produced immediately. Conversely, its rival, the discriminator network, is able to distinguish between samples derived from the generator framework and samples derived from actual data. Models for classification serve as the discriminator. The objective function of the GAN design is in Eq. (5), as stated in (Uddin, 2019).

$$arg \underset{Gm}{\min} \underset{Dm}{max} P(Dm, Gm) = E_{i\sim k_{data}(i)}\left[\log\left(Dm(i)\right)\right] + E_{a\sim k_a(a)}\left[\log\left(1 - Dm\left(Gm(a)\right)\right)\right] \quad (5)$$

where $Dm(i)$stands for the discriminator function. The likelihood that the input vector designated as i is from the training dataset is the output of this function. The $Dm(i)$ function returns a value between 0 and 1 when given an input of i.

Similar to this, $Gm(a)$is a generator function that, based on the z (noise vector), produces a matrix with the same dimensions as x. The distribution of chances of samples is represented as $k_{data}(i)$ and is taken from the training dataset. $k_a(a)$ stands for the sample distribution of chances from a noise generator.

The expectation function, abbreviated as $E(.)$, which is produced by the log-loss function as a positive class. In Eq. (6), the log loss function is defined.

$$E(s|\, r) = \frac{-1}{L}\sum_{x=1}^{L}(r_x(\log s_x) + (1 - r_x)(1 - s_x)) \quad (6)$$

The estimation is shown as s_x where the actual data is shown as r_x. The log function is employed when the model's response is anticipated to be 0 or 1. According to the probability distribution $s(i), E(f(i))$ of the given function $f(i)$

is expressed as in Eq. (7), when x is taken from $s(i)$.

$$E_{i\sim s}(f(i)) \int s(i)f(i)dx \quad (7)$$

In Eq. (5), there are two loops, such as $min_{GM}\, P(dm, gm)$ and $max_{DM}P(dm, gm)$. Maximizing the right side through discriminator parameter tweaking is the goal of $max_{DM}P(dm, gm)$. The objective function represented by Equation (5) has two loops that stand for $max_{DM}P(dm, gm)$ and $min_{GM}P(dm, gm)$. The goal of $min_{GM}P(dm, gm)$ is to minimize by adjusting the generator's parameters.

4. EXPERIMENTAL RESULT

The experiment using the proposed model on MMCD data is evaluated with respect to four significant metrics such as accuracy and f-measure, precision and recall. On a system with an Intel Core i7-7700k CPU, Hard Disk: 512GB, and 64 GB of RAM, we installed the suggested framework. The multimodal neural network algorithm must be accelerated by GPUs. Tensorflow and Python have been used to implement the ML algorithm and framework modules.

4.1 Bytes based result analysis and comparison

Table 3: Performance Result based on 1D Vector

Models	Precision	Recall	F measure	Accuracy
DBN	98	98.50	97.96	98.47
GAN	98.97	98.08	99.02	98.99
HCAGAN_DBN	99.07	99.12	99.46	99.61

The test results in Table 3 demonstrate that the 1D feature is superior to the 2D feature and that HCAGAN_DBN has a greater detection impact than DBN and GAN. The suggested 1D byte feature-based byte classifier achieves accuracy of 99.61%. The comparison result with other deep learning model like DBN and GAN and the graphical representation is shown in figure 6, 7 and figure 8.

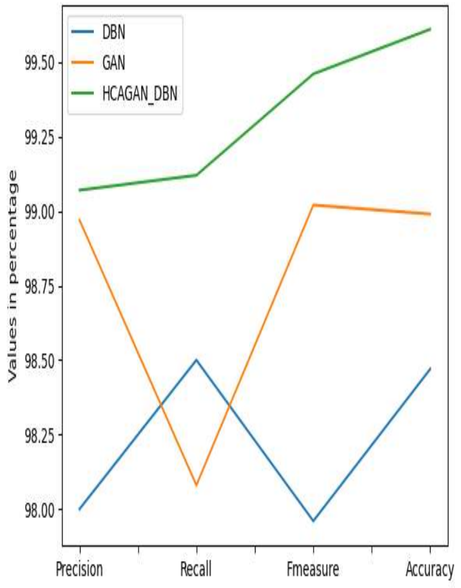


Figure 6: Performance Comparison with DBN, GAN and Proposed HCAGAN_DBN

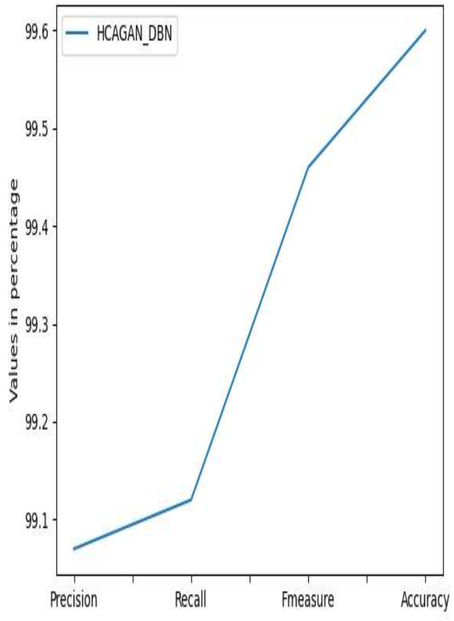


Figure 7: Performance Evaluation Result of Proposed Model for Byte Feature

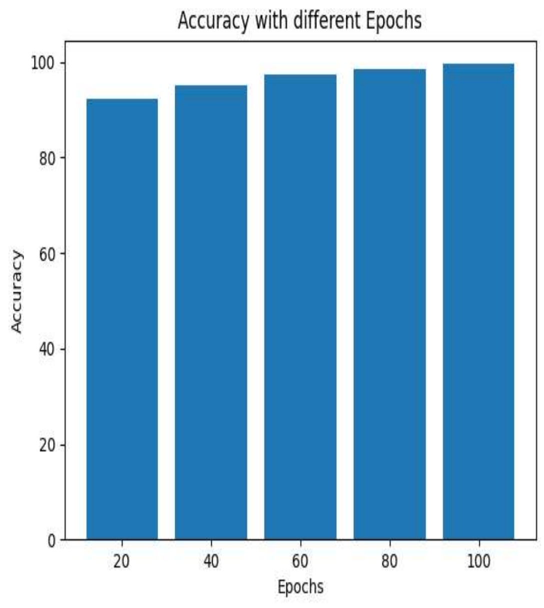


Figure 1: Accuracy Comparison of HCAGAN_DBN with different iterations

4.2 Opcode Based Result Analysis and Comparison

Sections lines, opcodes, stdcalls and n-grams are among the features retrieved from the training dataset. Table 4 provides the asm features' findings. The result is compared with other deep learning model like DBN and GAN and the graphical representation is shown in figure 9 and figure 10.

Table 4: Opcode Based Result Analysis

Models	Precision	Recall	F measure	Accuracy
DBN	97.82	98.3	97.64	97.76
GAN	98.12	98.7	99	97.92
HCAGAN_DBN	99.2	99.72	99.23	99.47

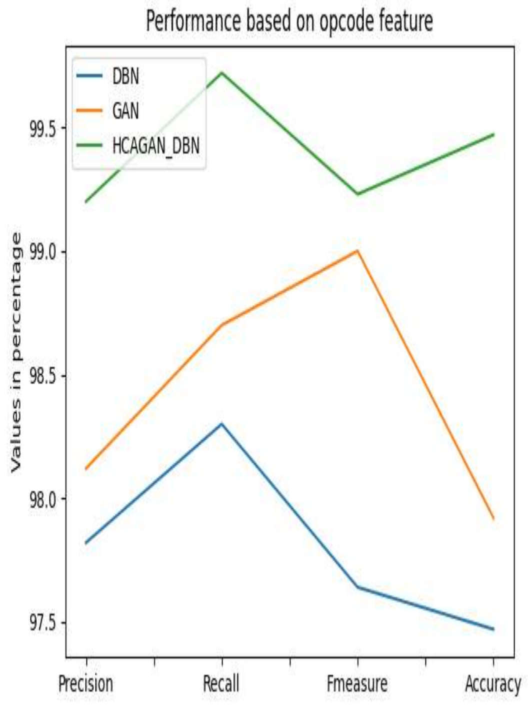


Figure 9: Performance Evaluation Result of Proposed Model

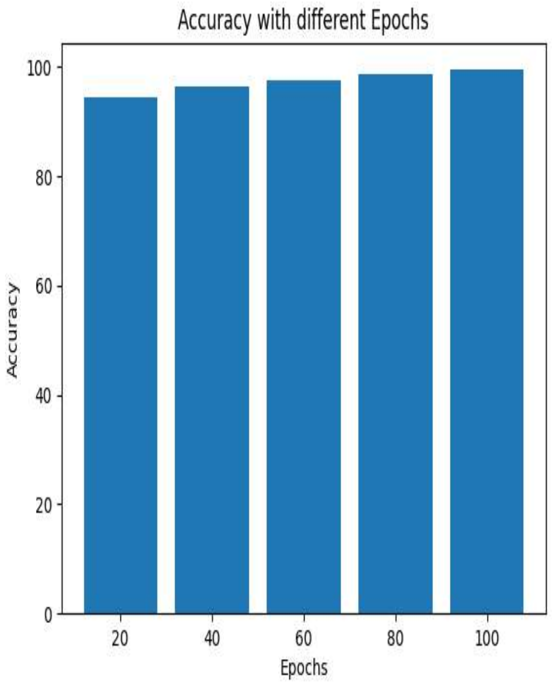


Figure 10: Accuracy comparison of HCAGAN_DBN with different iterations

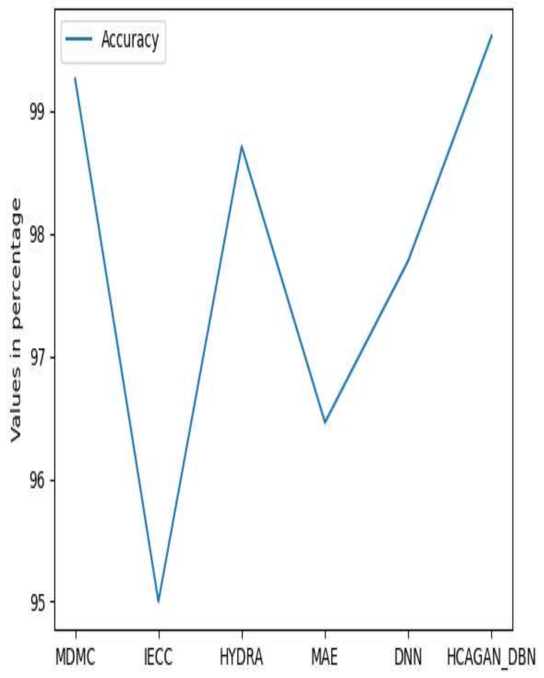


Figure 11: Accuracy Comparison with Existing models in Literature

We compared various related study findings with our findings, which are depicted in figure 11, in order to assess the efficacy of our methodology. They can be directly compared because the majority of studies in the literature used the same Microsoft malware dataset as we did.

Due to the increase of unknown malware volume needs the requirement of new hybrid deep learning algorithm for malware categorization. Hybrid deep learning algorithm provides more accuracy than simple deep learning algorithm. Hybrid algorithm achieves time complexity. Key contribution of our work is the utilization of a new hybrid model HCAGAN-DBN for classifying malware family with good performance result than the existing deep learning algorithms. New innovative layers are applied in the model to avoid over fitting. Analysis revealed that the asm classifier's 99.47% detection accuracy was higher than that of the other models in figure 11. The 1D vector feature achieved 99.61% accuracy which is higher than the approach.

5. CONCLUSION

An innovative deep learning approach for malware detection is put forth in this study. Experiments were used to confirm its performance. HCAGAN_DBN model was suggested in this study to categorize malware samples into malware families. The Microsoft Malware dataset was utilized to train the models using both conventional and hybrid deep learning techniques. To examine the malware feature, three different deep learning models such as GAN, DBN and HCAGAN-DBN are applied. HCAGAN-DBN shows the best performing classifier supporting in the outgoing efforts to combat the ever-growing threat land space. In the first phase the byte feature is extracted to 1D vector instead of converting gray scale image. Similarly the opcode is extracted from the asm feature and fed into the HCAGAN_DBN that perform well on both feature and achieved 99.61% and 99.47% accuracy respectively. Our study demonstrates multimodal deep learning for malware categorization. The proposed multimodal technique learns and mixes malware attributes from two information sources, improving classification performance when compared to classifiers that accept only one type of data as input. The improvement in the performance metrics highlights the impact of the proposed model in the area of malware analysis. Overall, HCAGAN-DBN offers considerable advancements in automatic detection of malware.

6. LIMITATIONS AND FUTURE WORK

The drawback of the work is not using of dynamic features. The model is applied with only one benchmark MMCC – BIG15 dataset and not with the two or more as well as real time datasets. Our future enhancement work is to analyze the malware with a real time datasets.

Conflicts of Interest

The authors declare that they have no conflict of interest.

Funding Statement

No funds, grants or other support was received.

AUTHOR CONTRIBUTIONS

Formal Analysis and Investigation: V.S. Jeyalakshmi, Conceptualization: V.S.Jeyalakshmi and Krishnan Nallaperumal, Resources: V.S.Jeyalakshmi, Validation: Krishnan Nallaperumal, Methodology: V.S.Jeyalakshmi, Writing Original Manuscript: V.S.Jeyalakshmi and Krishnan Nallaperumal, Supervision: Krishnan Nallaperumal.

REFERENCES:

[1] Kumar, Sachin, Prayag Tiwari, and Mikhail Zymbler, "Internet of Things is a revolutionary approach for future technology enhancement: a review" *Journal of Big data* 6, no. 1, 2019, pp. 1-21.

[2] G. Kambourakis, C. Kolias, and A. Stavrou, "The mirai botnet and the iot zombie armies" In *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM),* IEEE, October 2017, pp. 267-272.

[3] O. P. Samantray, and S. N. Tripathy, "An efficient hybrid approach for malware detection using frequent opcodes and API call sequences", In *Computational Intelligence: Select Proceedings of InCITe ,* Singapore: Springer Nature Singapore, *2022-2023,* pp. 727-735.

[4] M.A. Abdullah, Y. Yu, K. Adu, Y. Imrana, X. Wang, and J. Cai, "HCL-Classifier: CNN and LSTM based hybrid malware classifier for Internet of Things (IoT)", *Future Generation Computer Systems*, *142*, 2023, pp. 41-58.

[5] J. Singh, and K.K.K. Senapati, " Malware Analysis and Classification", In *Malware Analysis and Intrusion Detection in Cyber-Physical Systems, 2023,* IGI Global, pp. 42-63.

[6] T. Sarath, K. Brindha, and S.S. Senthilkumar, "Malware Forensics Analysis and Detection in Cyber Physical Systems", In *Malware Analysis and Intrusion Detection in Cyber-Physical Systems,* IGI Global, 2023, pp. 238-262.

[7] Yuan, Baoguo, Junfeng Wang, Dong Liu, Wen Guo, Peng Wu, and Xuhua Bao. "Byte-level malware classification based on markov images and deep learning", *Computers & Security,* 92, 2020, p. 101740.

[8] D. Gibert, C. Mateu, and J. Planes, " HYDRA: A multimodal deep learning framework for malware classification", *Computers & Security*, *95*, 2020, p.101873.

[9] Ö. Aslan, and A. A. Yilmaz, "A new malware classification framework based on deep learning algorithms", IEEE Access, 9, 2021, pp. 87936-87951.

[10] A. A. Darem, F. A. Ghaleb, A. A. Al-Hashmi, J.H. Abawajy, S.M. Alanazi, and A.Y. Al-Rezami, "An adaptive behavioral-based incremental batch learning malware variants detection model using concept drift detection and sequential deep learning", *IEEE Access*, *9*, 2021, pp. 97180-97196.

[11] R. Devi, Aiyshwariya, and A. R. Arunachalam. "Enhancement of IoT device security using an Improved Elliptic Curve Cryptography algorithm and malware detection utilizing deep LSTM", *High-Confidence Computing* 3, no. 2, 2023, p. 100117.

[12] M. N. Al-Andoli, K. S. Sim, S. C. Tan, P. Y. Goh, and C. P. Lim, "An Ensemble-Based Parallel Deep Learning Classifier with PSO-BP Optimization for Malware Detection", *IEEE Access, 2023.*

[13] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, M. Ahmadi, "Microsoft malware classification challenge", 2018.

[14] Microsoft Corporation. Microsoft Portable Executable and Common Object File Format Specification. Revision 9.3 – December 29, 2015. [Available: http://www. microsoft.com/whdc/ system/ platform/firmware/PECOFF.mspx].

[15] A. R. Mohamed, G. Dahl, and G. Hinton, "Deep belief networks for phone recognition", In *Nips workshop on deep learning for speech recognition and*

related applications, Vol. 1, No. 9, December 2009, p. 39.

[16] T. Li, J. Zhang, and Y. Zhang, "Classification of hyper-spectral image based on deep belief networks", In *2014 IEEE international conference on image processing (ICIP), IEEE, October, 2014,* pp. 5132-5136.

[17] S. Jeon, and J. Moon, "Malware –detection method with a convolutional recurrent neural network using opcode sequences", Information Sciences, 535, 2020, pp.1-15.