Hybrid Tensor Flow DNN and Modified Residual Network Approach for Cyber Security Threats Detection in Internet of Things

Abdulrahman Mohammed Alshehri[1†,*] and Mohammed Saeed Fenais[2††],

Riyadh Schools, Riyadh, kingdom of Saudi Arabia

Summary
The prominence of IoTs (Internet of Things) and exponential advancement of computer networks has resulted in massive essential applications. Recognizing various cyber-attacks or anomalies in networks and establishing effective intrusion recognition systems are becoming increasingly vital to current security. MLTs (Machine Learning Techniques) can be developed for such data-driven intelligent recognition systems. Researchers have employed a TFDNNs (Tensor Flow Deep Neural Networks) and DCNNs (Deep Convolution Neural Networks) to recognize pirated software and malwares efficiently. However, tuning the amount of neurons in multiple layers with activation functions leads to learning error rates, degrading classifier's reliability. HTFDNNs (Hybrid tensor flow DNNs) and MRNs (Modified Residual Networks) or Resnet CNNs were presented to recognize software piracy and malwares. This study proposes HTFDNNs to identify stolen software starting with plagiarized source codes. This work uses Tokens and weights for filtering noises while focusing on token's for identifying source code thefts. DLTs (Deep learning techniques) are then used to detect plagiarized sources. Data from Google Code Jam is used for finding software piracy. MRNs visualize colour images for identifying harms in networks using IoTs. Malware samples of Maling dataset is used for tests in this work.

Keywords:
IoT, malware detection, software piracy, cyber security, data mining, Tensor flow DNN.

1. Introduction

Malware's exponential growth has recently posed a serious cyber security threat. The number of new malware variants has surpassed a billion in the last three years, according to the Symantec internet security threat report [1]. Hackers are increasingly using techniques like packing and encryption to create new malware variants. Thus, rapid and accurate malware variant identification can help cyber security professionals understand their harmfulness and other attributes, which is valuable research [2]. The impacts of cyber risks on: Public/private institutional information systems; goodwill and stakeholder confidences are being increasingly felt. Cyber attacks break into corporate IT infrastructures for disrupt businesses or amusements or challenges. Websites or servers go down in cyber challenges between professionals. [3]. To reclaim service, access, or websites, current cyber threats and attackers require payment from the victimised enterprise.

Insurance policies were designed for protecting businesses against financial consequences due to threats on the Internet. Organizations specializing in risk assumptions and pooling foresaw financial opportunities in this area [4]. Many insurances exist now for cyber security breaches. When these organizations were new to this domain, it was difficult for them to collect data about electronic losses, but they have educated themselves on pricing policies and predicting potential losses. IoTs use electronic chips, sensors, and other moving hardware gadgets to get connected physically called "Things" to the internet. Devices in IoTs are identified by RFIDs (radio frequency identifiers) which can be controlled and monitored from distances [5]. IoTs connect intelligent physical objects, services, cloud computers, and applications irrespective of locations. IBM had predicted 50 billion internet connected devices [6]. Technology enables IOTs can be used to build smart cities, academics, e-commerce/banking, healthcare, industries, entertainments, and protections of human being to name a few of their applications. Networks connecting IoT devices can be attacked by Malwares while pirated software target industrial cloud IoTs thus compromising security.

Software piracy has emerged as a major issue of the software industry. It can be defined as unauthorized usage of complete software or portions without any legal license and agreement. Most individual uses of software are unaware that pirating software is a severe felony committed. Preventing software piracies are critical for the industry's growth as developments in real-world applications consume significant amounts of efforts in terms of development ideas and executions. Many approaches have been employed in this fight against software piracy. Software piracy is the illegal reuse of other people's source codes to create software that looks like the original. The cracker may reverse engineer the original software's logic and design it in another type of source code [7]. It is rapidly increasing and costs the software industry a lot of money [8]. According to the Business Software Alliance (BSA) 2016

Manuscript received October 5, 2022
Manuscript revised October 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.10.31

report, public software piracy costs businesses up to $52.2 billion annually. Many studies have shown that anywhere from 5% to 20% of all software contains plagiarised source codes. Intelligent software plagiarism detection is required to detect pirated software source code. Researchers have proposed multiple approaches for identifying software birthmarks which are based on design features and appropriate estimations of procedures copied. Though studies have thoroughly examined programmes in several perspectives of software piracy and theft detections, lack of objective metrics for software comparisons in terms of infringement detections are very less. An analysis of source code structure includes syntax trees, graph behaviour, and subroutine function call graphs [9]. This makes it difficult to detect if a cracker reuses the original software's logic in a different programming language. The industrial IoT cloud services can safeguard smart devices by detecting software plagiarism and malware[10]. Currently, malware recognition methods fall into two categories: static feature-based and dynamic feature-based. Static malware detection analyses the malware's raw bytes or disassembles it to analyse its opcodes, file structure, and other file attributes.

Attacks on networked IoTs are becoming easier as these networks grow where Malwares use internet to target nodes, computers, and smart phones. Many techniques have been proposed for detecting Windows based malwares where these methods can be dynamic or static. Dynamic approaches use code executions in virtual environments to learn malware patterns [11]. Suspicious behaviours can be identified by examining function calls or exploring parameters or data flows or tracing instructions or visual analyses of codes. Learning approaches use activation function's error rates to adjust neuron counts in layers where high errors result in degraded classifier performances. This research work proposes HTFDNNs and MRNs or Resnet CNNs for detecting software piracy and malwares.

The remaining of the research work is: Section 2 review the recent techniques for protecting the malicious attacks using advanced MLTs. section 3 briefs the proposed methodology. section 4 briefs the results and its explanation. section 5 deals with the conclusion and future work.

2. Literature Review

In this part, some recent cyber security techniques that work better when combined than when run separately. Musman et al. [12] implemented Cyber Security Game for determining best budgetary security measures. Their implementation considered cyber event's risk level and chances of threats. When more systems are interconnected all possible attack paths were defended in the Game, but attackers need only one path to succeed. The implementation minimized maximum cyber risk using MiniMax game theory. The study demonstrated their Cyber Security Game's methods using a point-of-sale system. Fielder et al. [13] introduced a new control game in which the security manager must pick between multiple levels of implementation of cyber security control, and a commodity attacker must choose between multiple targets to attack. We created a decision-making tool and a case study based on existing regulatory standards to compare these techniques. Feng et al. [14] developed innovative cyber risk managements for blockchain-based services. The scheme adopted cost-effective cyber insurances for reducing cyber risks on blockchain networks and assuming blockchain services included infrastructure providers, blockchain providers, insurers, and users. To keep blockchains on, suppliers bought computing resources with consensus from infra-framework suppliers like clouds before releasing blockchain services to customers. For modelling and evaluating distributed computing investments, Dhamal et al. [15] used stochastic game frameworks using dynamic players and where players were rewarded for overcoming problems or contributing computational resources but paid the cost of computational power and time consumed for usage like blockchain mining. On the other hand, players with cost parameters below a specific threshold invest maximal power in Markov perfect equilibrium. Players don't need to know the system state. As in volunteer computing, a payment for contributing to a common central entity's processing capacity is then considered. We use simulations to investigate the impact of players' entrance and departure rates on utilities in both scenarios. Kim et al. [16] developed a new Bitcoin mining protocol along with payment incentives applied on peer-to-peer groups in distributed computation network systems. Their investigations maximized Bitcoin users' benefits while using their cooperative game paradigm.

The presented technique is compared to other current schemes using system-level simulations. According to the simulation results, this solution exceeds existing Bitcoin schemes in terms of fairness and efficiency. Shomer et al. [17] studied the success rate of block-hiding mining in Bitcoin-like networks. Find two block-hiding techniques that outperform the typical mining strategy in this parameter space. This study recommends relative hashing power to measure a miner's influence on the network. But only when this measure of impact crosses a particular threshold. Grover et al. [18] presented CNNs and ANNsto recognize malware risks in data files across IoT data networks. While the CNN-based paradigm recognizes malware via pictures, the ANN-based paradigm distinguishes between malware and ordinary via incoming data traffic. Naeem et al. [19] developed MTHS (Malware Assault Hunting System) that transformed malware binaries into colour images for efficient malware recognitions. The study created baseline

accuracy comparisons of MTHS with traditional malware detection methods. Their scheme was tested on Windows and Android software datasets where their MTHS performed better in terms of reaction times and detection accuracies, outperforming MLTs and DLTs. Aslan et al. [20] introduced a new hybrid architecture that integrated pre-trained networks optimally. The study collected data where their designed DNNs were trained on evaluated on Malimg, BIG 2015, and Malevis datasets. The study's findings showed that their suggested procedure accurately classified malwares while outperforming other methods and scored 97.78% accuracy on the Malimg dataset exceeding most malware detection techniques based on MLTs. Bozkir et al. [21] examined modern convolutional neural networks (Resnet, Inception, DenseNet, VGG, AlexNet) with paradigm generation and inference time. A unique malware data set with 8750 training and 3644 test examples across 25 classes was also recommended and utilized. Using DenseNet networks, the maximum accuracy was 97.48 % using 3-channel (RGB) images. Ullah et al. [22] used a combined scheme where DLTs recognized pirated software and malwares- in networked Iots while TFDNNs identified pirated applications. Then deep learning recognizes source code plagiarism.. The DCNN also recognizes harmful infections in IoT networks using color images. The testing findings show that the presented technique to find IoT cybersecurity threats performs better than existing procedures. The existing system offers some benefits and drawbacks based on the preceding study. Unknown malware detection is a significant challenge.

3. Proposed Methodology

This paper suggests a paradigm of safety for industrial IoT's cyber security threats as depicted in Figure 1. The malware binaries and pirated software files are processed using 4 cloud storage databases. The first database contains raw network traffic data, while the second database contains a list of previous virus data, and the third database contains new virus signatures. Cracker stores stole software in IoT devices in database 4. It acts as a repository for unauthorized copies that crackers try to propagate via IoT networks. This study developed a HTFDNN and Resnet CNN-based technique for recognizing software piracy and malware. We present the Hybrid Tensor Flow DNN to recognize stolen software, starting with source code plagiarism. To reduce source code plagiarism, tokenization and weighting procedures filter out noisy data [23]. The MRN also visualizes color images to recognize harmful infestations in IoT networks. The first database fed the preprocessing module raw data. Raw data is preprocessed for valuable characteristics. The preprocessed data is sent to the recognizing module. The detection module learns from signatures in databases two and four to recognize malware and pirated applications. The suggested system alerts the administrator if malicious behavior is seen in the network.The primary objectives of this presented techniqueare.

- Large-scale malware recognitions with higher accuracies
- Software similarity recognitions from multiple programmers
- identifications of pirated copies of actual software and malwares in minimal computational costs

3.1. Malware Threat Detection

Traditional techniques may address code obfuscation concerns; however, texture feature mining utilizing virus visualizations has a high computational cost. Furthermore, these feature retrieval strategies do not work well with large volume of malware data. Malware is constantly being created, updated, and manipulated, making detection more difficult. The recommended malware detection strategy attempts to address all of the difficulties. This research offered a mixed deep learning strategy for recognizing pirated and Malware threats on the industrial IoT cloud. The Tensor Flow DNN is intended to recognize pirated software by plagiarising source code. The recommended strategy's combined solutions are pretty encouraging in terms of classification performance.

1) Data Preprocessing

To translate virus detection challenges as image categorization challenges, colour images are created from raw binary files which distinguishes the proposed work from other methods. Malware binaries are converted into colour equivalents instead of gray scale 256 colours for recovering more characteristics. Moreover, enhanced malware image characteristics perform well in malware family classifications [24]. Previously, multiple virus detection systems based on MLTs produced superior results by using grayscale images which were visualized, features retrieved and virus types classified. The categorization reliability is improved by applying feature reduction procedures to reduce the amount of characteristics in the collection. Deep learning procedures outperform large malware datasets because these procedures may employ filters to reduce noise automatically. As a result, employing color images yields superior outcomes when using DLTs. Malware binary files are converted to colour images through four stages. The raw binary files are first converted to hexadecimal strings (0-15) followed by 8-bit segment hexadecimal streams, measured as unsigned integers between 0 and 255. These 8-bit vectors are then converted to 2D matrices and each 8-bit integer generated in these 2D

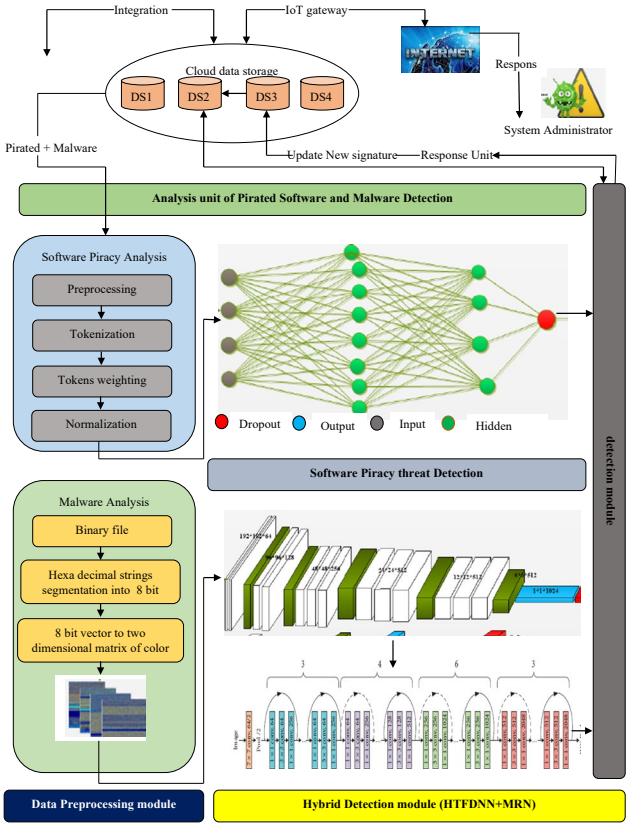

Figure 1. The process of the proposed methodology

matrices are represented by shades of red, green, and blue colours. Data pre-processing steps are depicted in Figure 1.

2) DCNNs (Deep CNNs)

DCNNs were established for thorough malware data analyses and consists of 5 modules (refer Figure 1). Training images are fed to the input layer. First, a convolution layer reduces noise and improves signal qualities. A pooling layer reduces data overhead while retaining important information [25]. Finally, a entirely linked layer is utilized to transform the 2D array to a 1D array, then fed into the appropriate classifier. Fourth, the classifier is utilized to recognize malware families from respective images.

3) Convolution Layer

The key characteristics are acquired by applying a convolution layer to reduce the image of attributes. Convolution layers through interpretations, rotations, and scale invariance minimize issuer of over fits introducing generalizations in basic designs. As indicated in equation 1, the convolutional layer's input collects maps.

$$x_j^l = f(\sum_{i \in M_j} x_j^{l-1} * k_{ij}^l + b_j^l) \quad (1)$$

M_jindicates given map clusters; k_ij^l stands for convolution kernel combining ith and jth input feature maps while b_j^l represents consistent bias of ith feature map, and f represents the activation function.

4) Pooling Layer

Pooling layers execute maximum or average pooling and called as sub-sampling layers. They are used to reduce the visual distortion effects and are typically unaffected in backward propagations. Also the characteristics are reduced while enhancing recommended DCNN functionalities. Equation (2) demonstrates the aforesaid descriptions.

$$x_j^l = f(down(x_j^{l-1}) + b_j^l) \quad (2)$$

In which down (.) performs a pooling task, and b represents biasvalue.

Fully Connected Layer

It categorizes the outcome of pooling layer.Each neuron communicates with the neuron before it via this layer. It is designed to increase paradigm generalization capability by reducing over fits. Noises are eliminated by the use of filters on original data during fine-tuning where the count and sizes of kernels improve signal characteristics.

Learning

Malware samples are grouped according to their family or class. Equation (3) depicts losses while training data.

$$Loss = -\log\left(\frac{exp(f_{zt})}{\sum_k exp(f_{zt})}\right) \quad (3)$$

In whichf_ztrepresents the rank for the kth class; and f_ztindicates the score for the appropriate family. The attributes of the model are learned using Adam optimizer to lower the loss that occurred on the training data.

3.2. Software Piracy Threat Detection

Unlawful copying or dissemination of copyrighted software aptly define software piracies which have no legal meaning. Software piracies are types of copyright infringements that occur in higher education settings where peer's knowledge is exchanged. Software piracies can be prosecuted under copyright infringement statutes. DLTs can play major roles in identifying pirated software compiled from multiple source codes. As demonstrated in Figure 1, plagiarised version of the programme is a pirated copy of software where crackers have copied original software's logic. Pre-processing methods are begun by tokenizing source codes to reduce data's dimensions and acquire important features that can be utilised to identify plagiarism even when split across various source codes.

1) Pre-Processing and Feature Extraction

Since recognizing source code's syntax and semantics is complex, plagiarism detection techniques were applied to trace software thefts. The methods use stemming root words and frequency retrievals. They decode codes and filter out noises. Unwanted information is removed, such as special symbols, constants, and stop words. Tokenization is then utilized to turn the cleansed data into useable tokens. To mine more valuable information, pre-processing processes employ stemming of root words, and limitations

on frequency of words. Following that, the contribution of each token is weighed. In the weighting phase, the TFIDF and Log TF are used. In Equation 4, TFIDF is defined.

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D) \quad (4)$$

In which t indicates token, f indicates the number of frequency, d represent every individual document, and D indicates all documents in the dataset.

2) Hybrid Tensor Flow DNN And MRN

The effectiveness of MLTs can be determined based on its learning of suitable data representations called representative learning. Traditional MLTs demand task specific data extractions using preset techniques. DLTs, on the other hand, depend on neural networks to learn effective feature representations automatically through nonlinear transformations of raw data characteristics like word vectors or image pixels. DLTs train themselves based on objective functions which implant model's aims i.e. they successfully extract salient features that are useful for goals without models being knowledgeable in the domain. This work acquires harmful code image characteristics based on a deep network paradigm. Deep learning networks have varied architectures and operate differently. However, as network depth increases, the deep learning network will encounter various drawbacks, such as the degradation challenge, which means that adding more layers to a sufficiently deep network would result in more training mistakes. However, the ResNet network addresses these drawbacks. Furthermore, these deep networks offer numerous advantages and have outperformed the existing network in image detection. We assess the reliability of these deep networks and then acquire byte sequence binary image characteristics using the hybrid tensor flow DNN and modified ResNet50 deep learning networks.

Deep Learning With Tensor flow Framework

Tensor Flow is a ML system for high-level calculations in a complicated environment. And use Tensor Flow's API to implement many machines and deep learning procedures. It characteristics many sorts of layers that may be designed for sophisticated computations, data training, and supervising the state-run of each function. Using the Tensor Flow framework, the in-depth learning strategy aims to recognize related source codes in many programming languages. The acquired similar principles are then utilized to recognize the pirated program. The weighting values are fed into the deep learning paradigm as input. Data inputs and outputs are built up in dense layers also called absolutely linked layers. Three thick layers of 100, 50, and 30 neurons have been used in this work. Data is received by the first layer through input variables with input shape parameters. Plagiarised code is targeted by dense layers which employ resultant variables. Equation (5) depicts positive portions of arguments.

$$f(x) = x^{+} = \max(0, x) \quad (5)$$

In which x indicates the input to the equivalent neurons, and sigmoid is described in equation 6.

$$S(x) = \frac{1}{1+e^{-x}} \quad (6)$$

In which S indicates the sigmoid function. The Adam optimizer, also known as the stochastic descent gradient, is employed in the compilation and optimization of deep learning paradigms. For each limitation, it evaluates the discrete adaptive learning rates. For example, equations 7 and 8 show the decaying means of past squared gradients.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \quad (7)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g^2 \quad (8)$$

In which m_tand v_t stand for the first and second instant gradients with predictable meanings. The exponential running average of the first gradient m_t and square gradient v_t are updated using estimates. The use of tensor flows in this work contributes in the following ways: It combines many types of computational APIs, such as the GitHub framework, to create and enhance MLTs for substantial data sets.

· This framework automatically trains the system using input, hidden, and outcome layers with varying activation functions.

· It supplies dependable services while also upgrading and expanding the recommended framework.

While deep learning techniques have shown promise in image detection, they are limited by two drawbacks: vanishing/exploding gradients and degradation. Normalized initialization and intermediate normalization layers have addressed vanishing/exploding gradients. But the deep learning network has another difficulty: degradation. The accuracy will progressively climb to saturation and then rapidly fall. But the issue isn't oversizing. A network with more layers (degraded network) has a more significant mistake rate in the equal training round. But the residual network could help with degradation. The following section describes the MRN or Resnet CNN process.

MRN or Resnet CNN

The residual unit is introduced in the residual network, and its framework is depicted in Figure 2. The residual mapping is established in the residual network as follows:

$$H(x) = F(x) + x \quad (9)$$

In which F(x) is the residual function. The new residual unit connection strategy is a shortcut connection which is specified in the residual network as follows:

$$y = F(x, W_i) + x \quad (10)$$

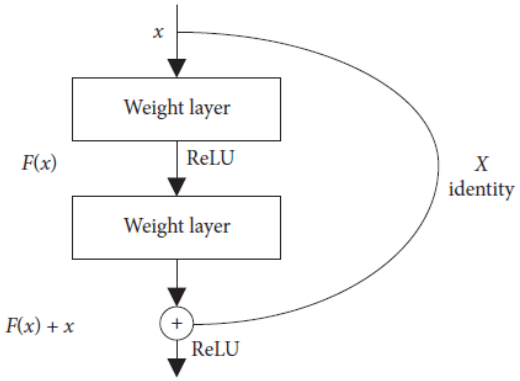

Figure 2: The residual unit.

In which F(x,W_i) denotes to=be-learned residual mapping to be learned. For the example in Figure 2, the F(x,W_i)in which σ indicates ReLU, F + x indicates the shortcut connection. If the dimensions of X and F(x) are varied in the deep learning network, is described as the below:

$$y = F(x, W_i) + W_s x \quad (11)$$

where W_sis a linear projection of the x to make it consistent with F(x) dimension, the degradation difficulty is reduced by importing residual data and residual units into the network. We ran many tests to assess the Reset50 network's reliability. The results demonstrate that ResNet50 can acquire byte sequence color picture characteristics. Figure 3 depicts the ResNet50 network framework. ResNet50 has 50 layers and 3.8 × 109 parameters [26].

The byte sequence level binary picture visualization strategy takes the malicious code's raw byte data as the image pixels' original data and arranges them in byte order, disregarding their position information. This section proposes an image conversion strategy including position data. The malware's raw bytes contain two types of data: location and pixel. The transformed binary image could then collect more information to improve harmful code categorization. The malware visualization in this section is also based on raw bytes, which raises two drawbacks: first, how to transform dangerous code data into pixel data, and second, how to select the image format. The malware's raw byte is still considered 8-bit unsigned integer data, and every five bytes is a group. Each data group contains two bytes of position data and three-pixel information.

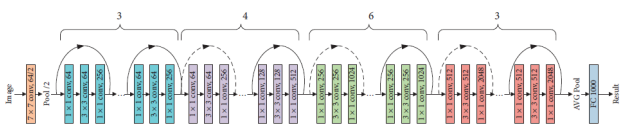

Figure 3. The structure of the ResNet50 network.

Each byte's value range is [0, 255], and each image is 256 ∗ 256. The image's bottom left data coordinates are set to (0, 0). The image's pixels are all set to 0. The first byte of each data group represents the pixel's X coordinate, the second the Y coordinate, and the final three the pixel's values. Then each set of data becomes a pixel in the image. If the data exceed 255, the 255 remainder procedure is applied. The binary image is 256 ∗ 256 and does not require standardization. This strategy allows the deep learning network to perform better at the equal depth.

Entropy sequence formation based data distribution

Entropy is a range of data distribution randomness and uncertainty. Divide the malware's raw bytes into continuous data blocks (00h-FFh in hexadecimal), evaluate the entropy of each block, and finally, connect the entropy of each block according to the block order to produce the entropy sequence. Each byte's value range is [0, 255]. The block's data must be utilized to determine the entropy value. So the block size is 256. If the last block is less than 128 bytes long, it is discarded from the entropy sequence. If not, a zero is added to the block to make it 256. For each block, evaluate entropy as follows:

$$H(x) = -\sum_{i=0}^{255} p(x_i) * log2p(x_i) \quad (12)$$

In which xi indicates a particular raw byte value and pi indicates the frequency of this value in the block, H(x) indicates the entropy value of the block, and the range of its value is zero to eight. Entropy is zero when all bytes in a block are equal. Entropy is eight if all values in the block are different. The entropy sequence is Hs = h1, h2, hn for a single raw byte divided into N blocks. A hybrid paradigm effectively recognizes malware threats.

Results and Discussion

A prototype system was built to test the effectiveness of the presented technique, and is developed in Java. This study employs a 10-fold cross-validation strategy to ensure the accuracy and reliability of the test findings. The presented work employs measurements such as Accuracy, Recall, Precision, and F-measure.

Precision is characterised as the proportion of accurately recognised positive observations to all anticipated positive observations.

Precision = TP/(TP+FP) (13)

The proportion of recognised positive observations to total observations is described as recall.

Recall = TP/(TP+FN) (14)

The F1 score is the weighted average of Precision and Recall. Therefore, it requires both false positives and false negatives.

F1 Score = 2*(Recall * Precision) / (Recall + Precision) (15)

Accuracy is measured in positives and negatives, as shown below:

Accuracy = (TP+FP)/(TP+TN+FP+FN) (16)

4.1. Performance Evaluation of Malware Detection

The influence is measured on the basis of classification performance for various malware image ratios. The image ratios are 224*224 and 229*229. The Leopard Mobile dataset has 14,733 malicious and 2486 benign samples, and the 229*229 ratio outperforms the 224*224 ratio. The classification accuracy of 229*229 and 224*224 images differs significantly. So, the 229*229 image ratio is a good choice for the presented malware recognition method. This study compares the outcomes of multiple image ratios. The Leopard Mobile dataset's 229*229 image ratio achieved 98.14 percent testing accuracy in 34s.

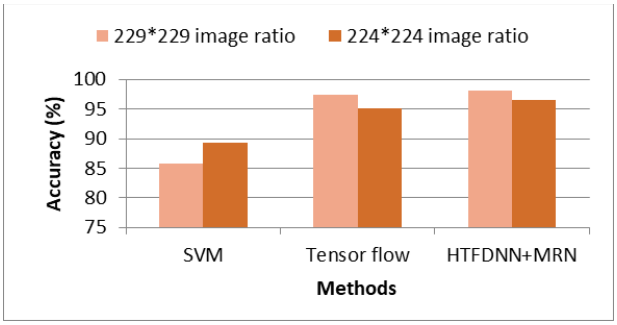

Figure 4. Accuracy comparison between the presented and traditional ML methods for malware detection

Figure 4 depicts the accuracy comparison between the presented and traditional malware detection methods. The graph shows that these methods have been evaluated with two types of image ratio pixels (229*229 vs 224*224). When the image ratio is 229*229, the proposed method HTFDNN+MRN has 98.14 percent accuracy, compared to 97.46 percent for Tensor Flow and 85.71 percent for SVM. When the image ratio is 224*224, the presented HTFDNN+MRN has 96.54% accuracy, while the Tensor flow approach has 95.1% and the SVM has 89.24%. For image ratio 229*229, the presented technique has high accuracy rate.

Figure 5. Precision comparison between the proposed and existing MLTs for detecting malwares

Figure 5 compares the precision of presented and traditional techniques. The malware dataset uses two classes: malicious and benign. Overall, 90% of classes are recognized for malicious files, with 10% missing. The HTFDNN+MRN technique outperforms the traditional methods in terms of precision.

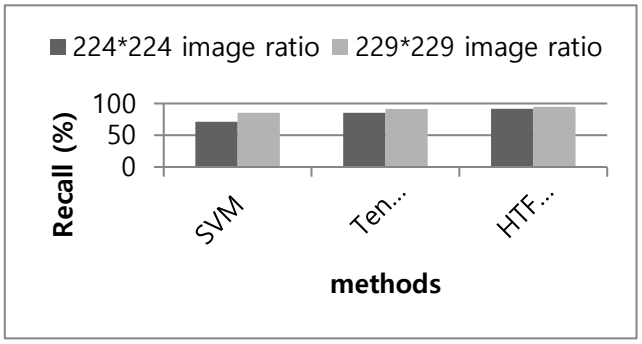

Figure 6. Comparison result of recall between the proposed and existing ML methods for malware detection

Figure 6 shows the recall comparison between the proposed and traditional malware detection methods. As shown in the figure, the presented technique outperforms SVM and Tensor flow techniques in terms of recall. For 229*229 images, the HTFDNN+MRN has a recall rate of 94.68%, while the Tensor flow and SVM has 91.47% and 85.24%, respectively. When the image ratio is 224*224, the HTFDNN+MRN has 91.67% recall rate, while the Tensor flow and SVM has 85.29% and 71.24%, respectively So this technique has high recall rate for 229*229 image ratio.

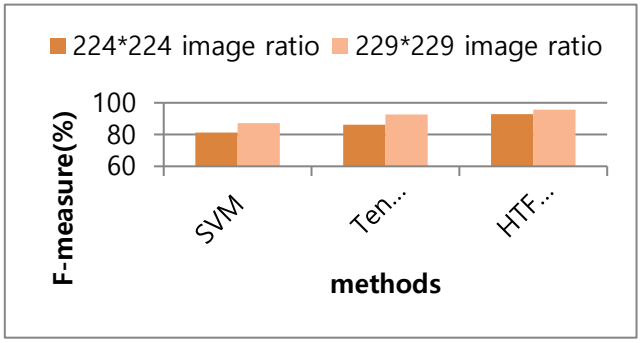

Figure 7. Comparison result of F-measure between the proposed and existing MLTs in malware detection

Figure 7 shows the f-measure comparison between the presented and traditional malware detection techniques. The figure shows that the proposed method outperforms existing SVM and Tensor flow methods in terms of recall. When the image ratio is 229*229, the HTFDNN+MRN has a high f-measure rate of 95.67%, compared to 92.68% for Tensor flow and 87.25 percent for SVM. When the image ratio is 224*224, this presented technique has 92.84% f-measure rate, while the Tensor flow and SVM have 86.24% and 81.27 %, respectively. With 229*229 image ratio, this suggested approach has high f-measure rate.

4.2.Performance Evaluation for Software Piracy

The software plagiarism measure examines pirated software's code. The presented software piracy technique was tested on GCJ (Google Code Jam) source code dataset. Initializations were extracting token from sources with details on frequencies. Subsequently, TFIDF (term frequency and inverse document frequency) and LogTF (logarithm term frequency) were used to select and extract token weights. Each token's contribution is weighted differently in the document or across all documents.

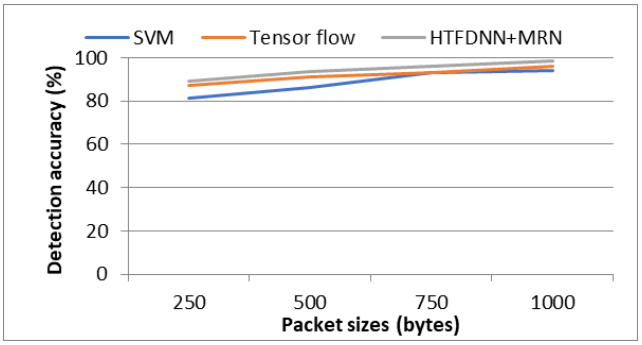

Figure 8. Detection accuracy of the proposed HTFDNN+MRN for piracy software detection

Figure 8 shows that the presented HTFDNN+MRN outperforms existing SVM and tensor flow based techniques with 98.24% detection accuracy for 1000 byte packets. Thus, the HTFDNN+MRN outperforms existing algorithms in terms of detecting results with high accuracy. The suggested learning techniques are robust to training data noise and thus achieve better accuracy rates while avoiding local optima. These results confirm that the described technique can recognize piracy software more effectively and reliably.

Conclusion

The key obstacles in cybersecurity using IoT-based big data are detecting software piracy and malware threats. We present a hybrid Tensor flow DNN with modified Residual Network approach for detecting pirated and malicious files. First, a Tensor Flow neural network is presented to recognize software plagiarism. To test the proposed approach, 100 GCJ programmers' source code files were collected. Code sources are pre-processed by eliminating noises and capturing useful tokens. Feature's contributions are assigned weights based on TFIDF and LogTF weighting strategies. The weighting values are then fed into the deep learning algorithm. To detect malware using IoT, a novel hybrid approach and colour image visualisation methodology is proposed. Current SVM and tensor flow based software piracy detection techniques are outperformed by 98%. The HTFDNN+MRN technique has 98.14 %, Tensor flow has 97.46 %, and SVM has 85.71 % when the image ratio is 229*229. Based on the test findings, the presented scheme is superior to other techniques in distinguishing malware variants of related families. The visual characteristics of malware are critical to future malware classification performance. Exploit entropy feature extraction and present reliable malware classification techniques.

REFERENCES

[1] Feng, S., Xiong, Z., Niyato, D., Wang, P., Wang, S. S., & Zhang, Y. (2018, December). Cyber risk management with risk aware cyber-insurance in blockchain networks. In 2018 IEEE Global Communications Conference (GLOBECOM) (pp. 1-7). IEEE.

[2] Ma, X., Guo, S., Li, H., Pan, Z., Qiu, J., Ding, Y., & Chen, F. (2019). How to make attention mechanisms more practical in malware classification. IEEE Access, 7, 155270-155280.

[3] Bozkir, A. S., Cankaya, A. O., & Aydos, M. (2019, April). Utilization and comparision of convolutional neural networks in malware recognition. In 2019 27th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.

[4] Sriram, S., Vinayakumar, R., Sowmya, V., Alazab, M., & Soman, K. P. (2020, July). Multi-scale learning based malware variant detection using spatial pyramid pooling network. In IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) (pp. 740-745). IEEE.

[5] Gupta, G. P., & Kulariya, M. (2016). A framework for fast and efficient cyber security network intrusion detection using apache spark. Procedia Computer Science, 93, 824-831.

[6] Li, J. H. (2018). Cyber security meets artificial intelligence: a survey. Frontiers of Information Technology & Electronic Engineering, 19(12), 1462-1474.

[7] Sabar, N. R., Yi, X., & Song, A. (2018). A bi-objective hyper-heuristic support vector machines for big data cyber-security. Ieee Access, 6, 10421-10431.

[8] Cronin, G. (2002). A taxonomy of methods for software piracy prevention. Department of Computer Science, University of Auckland, New Zealand, Tech. Rep.

[9] Djekic, P., & Loebbecke, C. (2005, July). Software piracy prevention through digital rights management systems. In Seventh IEEE International Conference on E-Commerce Technology (CEC'05) (pp. 504-507). IEEE.

[10] Mishra, B. K., Raghu, T. S., & Prasad, A. (2005). Strategic analysis of corporate software piracy prevention and detection. Journal of Organizational Computing and Electronic Commerce, 15(3), 223-252.

[11] Sharma, V. K., Rizvi, S. A. M., & Hussain, S. Z. (2010). Distributed Co-ordinator Model for Optimal Utilization of Software and Piracy Prevention. International Journal of Computer Science and Security (IJCSS), 3(6), 550.

[12] Musman, S., & Turner, A. (2018). A game theoretic approach to cyber security risk management. The Journal of Defense Modeling and Simulation, 15(2), 127-146.

[13] Fielder, A., Panaousis, E., Malacaria, P., Hankin, C., & Smeraldi, F. (2016). Decision support approaches for cyber security investment. Decision support systems, 86, 13-23.

[14] Feng, S., Wang, W., Xiong, Z., Niyato, D., Wang, P., & Wang, S. S. (2018). On cyber risk management of blockchain networks: A game theoretic approach. arXiv preprint arXiv:1804.10412.
[15] Dhamal, S., Ben-Ameur, W., Chahed, T., Altman, E., Sunny, A., & Poojary, S. (2018). A stochastic game framework for analyzing computational investment strategies in distributed computing. arXiv preprint arXiv:1809.03143.
[16] Kim, S. (2016). Group bargaining based bitcoin mining scheme using incentive payment process. Transactions on Emerging Telecommunications Technologies, 27(11), 1486-1495.
[17] Shomer, A. (2014). On the Phase Space of Block-Hiding Strategies. IACR Cryptol. ePrint Arch., 2014, 139.
[18] Grover, M., Sharma, N., Bhushan, B., Kaushik, I., & Khamparia, A. (2020). Malware Threat Analysis of IoT Devices Using Deep Learning Neural Network Methodologies. In Security and Trust Issues in Internet of Things (pp. 123-143). CRC Press.
[19] Naeem, H. (2019). Detection of malicious activities in internet of things environment based on binary visualization and machine intelligence. Wireless Personal Communications, 108(4), 2609-2629.
[20] Aslan, Ö., & YILMAZ, A. A. (2021). A New Malware Classification Framework Based on Deep Learning Algorithms. IEEE Access.
[21] Bozkir, A. S., Cankaya, A. O., & Aydos, M. (2019, April). Utilization and comparision of convolutional neural networks in malware recognition. In 2019 27th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.
[22] Ullah, F., Naeem, H., Jabbar, S., Khalid, S., Latif, M. A., Al-Turjman, F., & Mostarda, L. (2019). Cyber security threats detection in internet of things using deep learning approach. IEEE Access, 7, 124379-124389.
[23] Bandara, U., & Wijayrathna, G. (2012). Detection of source code plagiarism using machine learning approach. Int J Comput Theory Eng, 4(5), 674.
[24] Haddi, E., Liu, X., & Shi, Y. (2013). The role of text pre-processing in sentiment analysis. Procedia Computer Science, 17, 26-32.
[25] Liu, M., Shi, J., Li, Z., Li, C., Zhu, J., & Liu, S. (2016). Towards better analysis of deep convolutional neural networks. IEEE transactions on visualization and computer graphics, 23(1), 91-100.
[26] Hanif, M. S., & Bilal, M. (2020). Competitive residual neural network for image classification. ICT Express, 6(1), 28-37.

Abdulrahman Mohammed Alshehri is a Saudi MAWHIBA alumnus and a merit scholarship student recipient at Riyadh Schools. He is a Certified Ethical Hacker (CEH). Additionally, he has completed over ten certified courses, participated in numerous internships and clubs, and done volunteer and innovative work, such as his Telegram initiative channel. Furthermore, he is working on a personal Saudi cultural website whose goal is to have an impact worldwide. With all of this stated, he is enthusiastic about achieving the Kingdom's 2030 vision. His research interests include cybersecurity and technology.

Mohammed Saeed Fenais, a student at Riyadh Schools, has been presented with a distinction award in recognition of his outstanding academic achievement. He has more than 1000 hours in Olympiad training for competitive programming using C++ while maintaining a perfect grade point average at school. Furthermore, he is a MISK and Mawhiba alumnus. It is impossible not to get swept up in his contagious enthusiasm for bringing the Kingdom's vision for the year 2030 to fruition. His research interests include the development of new technology as well as the investigation of ways to keep computers secure..