International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Mechanized Detection and Extraction of Malware Using Deep Learning Approaches

V. S. Jeyalakshmi[1]. N. Krishnan [2]

Submitted: 05/05/2024 **Revised**: 18/06/2024 **Accepted**: 25/06/2024

Abstract: Malware creation is developing a considerable dangerous to the individuals as well as an organization. Protecting against these risks is continually being processed by the digital protection cyber specialists. The obscurity of categorizing malware is high since it might take many patterns and is continually evolving. With the support of artificial intelligence can undoubtedly access the large information, neural networks can be able to deal this problem very easily. This research aims to furnish effective by applying convolutional neural network with multi-layers to handle the situations of using imbalanced datasets. The proposed model developed by applying a Convolutional Neural Network with multi layers performed best to categorize the malware with 96.25% of accuracy. Generally, the malware classification problem is eased by the approach of converting it to binary images and then classifying the generated images.

Keywords: *Cyber security, Gray Level Run Length matrix, Artificial Intelligence, Deep Learning, Multi-Layer convolution neural networks.*

I. Introduction

Due to the scarcity of cyber security professionals remains a problem for the business organization to recognize their importance. The International Information Systems Security Certification Consortium [4] estimates the need of 2.72 million cyber security experts in world-wide. At the same time, malicious software with many thousands of new malware signatures is being discovered every day. Kaspersky's detection algorithm has discovered an average of 346,000 new malicious files daily in 2018, 360,000 in 2020 and its count still increases in 2023. Sixty-two percent of the malware samples were Trojans. There was an overall 40.5% rise in the number of Trojans found in comparison to the previous year's result [3]. So, malware analysis is an important task to provide security in traditionally manual fields. Malware analysis has often been performed using either static or dynamic malware analysis techniques or both [1, 2]. A new technique of memory-based analysis has been inspired by the appearance of file-less malware. A high-level explanation of malware analysis and its methodologies are explained here.

Static malware analysis may determine a file's signature is harmful or not without actually running the file. Several techniques are used to discover the malware signatures that identify a particular executable file as dangerous. Traditional static analysis techniques are very simple, will not find the malware signatures deeply. Here the advanced malware is more complicated to detect it by the use of code obfuscation methods. Static Malware Analysis is done by comparing the executable's hash file with the malware using tools like Virus Total. Strings and control flow on the executable file are analyzed by IDA disassemble tool to detect the malware.

Dynamic malware analysis is used to execute the malware file in a virtual environment to monitor the files behavior, network traffic activities by wireshark. Process monitor discovers how the malware interact with the host system by registry keys, etc.

Memory malware analysis is used to detect the advanced persistent threats, harmful code in file-less malware exists in the RAM of an infected machine to capture and dump a memory image. It was analyzed by the process lists and associated threads, networking information and interfaces (TCP/UDP), command history, system calls, kernel hooks, etc.

II. Related Works

This section provides a summary of earlier malware categorization algorithms based on images by deep learning algorithm with EXE file [14, 15]. Nataraj et al. (2011), a raw binary file's one-dimensional form is similar to its pictorial representation [13]. AI can speed up the investigation process without making any mistakes (James and Gladyshev, 2013). This study contributes a deep learning model that can accurately categorize raw binary files into one of 9 types of malware with an accuracy of 98.2 percent. Raff et al.'s (2017), the batch normalization model's inefficiency, the strategy slowed down the procedure of learning when applied to the study of binary executables in the batch normalization [7]. In

[1]*Research Scholar, Centre for Information Technology and Engineering, Manonmaniam Sundaranar University, Tirunelveli, India.*
vsjeyalakshmiap@gmail.com
[2]*Senior Professor (Retd.), Centre for Information Technology and Engineering, Manonmaniam Sundaranar University, Tirunelveli, India.*
krishnan17563@gmail.com

modern deep learning, batch normalization has been studied mostly in the context of image and signal processing, with natural language processing coming in a close second. There is a standard data structure across all of these fields. In contrast, the bytes in our binary data provide a new multi-modal nature, with byte values having vastly various meanings depending on the context, from plain ASCII text to code to structured data to images [10]. The working hypothesis is that this multi-modality results in a multiplicity of activation modes, which runs counter to the foundational principles of batch normalization leads to the degraded output. Research by Raff et al. (2017) supports this notion. Le et al. (2018) proposed a deep learning-based classification of malware strategy that relies only on data to identify complicated patterns and features, negating the need for any prior domain expertise. The study highlights how AI might help digital evidence of detecting malware. Le et al. (2018) a CNN-Bi-LSTM architecture to improve upon the performance of a CNN model by making use of the sequential representation [6, 8, 9, 11, 12].

III. Proposed Work

Convolutional neural network is conducted as static malware identification and classification experiments for testing the hypothesis. Multi-Layer convolutional neural network (MLCNN) designs were evaluated, compared for their efficacy in resolving this multi-classification task. Due to the computational complexity and the training time with limited resources in an IoT device, low powered edge devices give a gateway of IoT-based malware. There were 34 million IoT malware in 2019, according to the data collected by SonicWall Capture Labs, 2020 had increased to 66% and 400% growth of IoT malware attacks rise in 2023[5]. The effectiveness of each network design was evaluated using standard neural network performance indicators that include accuracy, loss, F1 scores, precision, and recall.

3.1 Dataset Explanation

In Malimg collection, the PNG images of 9,339 distinct malware binary files as portable executable (PE) files and 25 distinct types of malware in the dataset's images [13]. The sample sizes of malimg class are shown in Table 1.

During training, each CNN learn to recognize the characteristics of one malware class distinguishes from another for validation, CNN correctly predicts the malware class of a given binary image. The number of occurrences for each of the 25 classes is shown in Table 1.

Table 1: Sample sizes of the Malimg dataset

Malware Family	Type	Samples	Malware Family	Type	Samples
Adialer.C	**dialer**	**122**	**Lolyda.AA1**	**pws**	**213**
Agent.FYI	**bd**	**116**	**Lolyda.AA2**	**pws**	**184**
Allaple.A	**worm**	**2,949**	**Lolyda.AA3**	**pws**	**123**
Allaple.L	**worm**	**1,591**	**Lolyda. AT**	**pws**	**159**
Alueron. gen!J	**trojan**	**198**	**Malex. gen!J**	**trojan**	**136**
Autorun.K	**worm**	**106**	**Obfuscator. AD**	**dlr**	**142**
C2LOP.gen!g	**trojan**	**200**	**Rbot! gen**	**bd**	**158**
C2LOP.P	**trojan**	**146**	**Skintrim.N**	**trojan**	**80**
Dialplatform .B	**dialer**	**177**	**Swizzor.gen!E**	**dlr**	**128**
Dontovo.A	**dl**	**162**	**Swizzor.gen!I**	**dlr**	**132**
Fakerean	**rogue**	**381**	**VB.AT**	**worm**	**408**
Instantaccess	**dialer**	**431**	**Wintrim.BX**	**dlr**	**97**

Allaple. A and Allaple. L are roughly half of the malware samples in the dataset. SciKit Learn's library package was used to standardize all the training data's class weights before being used to train the neural networks. This may not be the best distribution for research, but it does return reality more closely. Figure 1 shows the histogram of the different categories in the Malimg dataset.

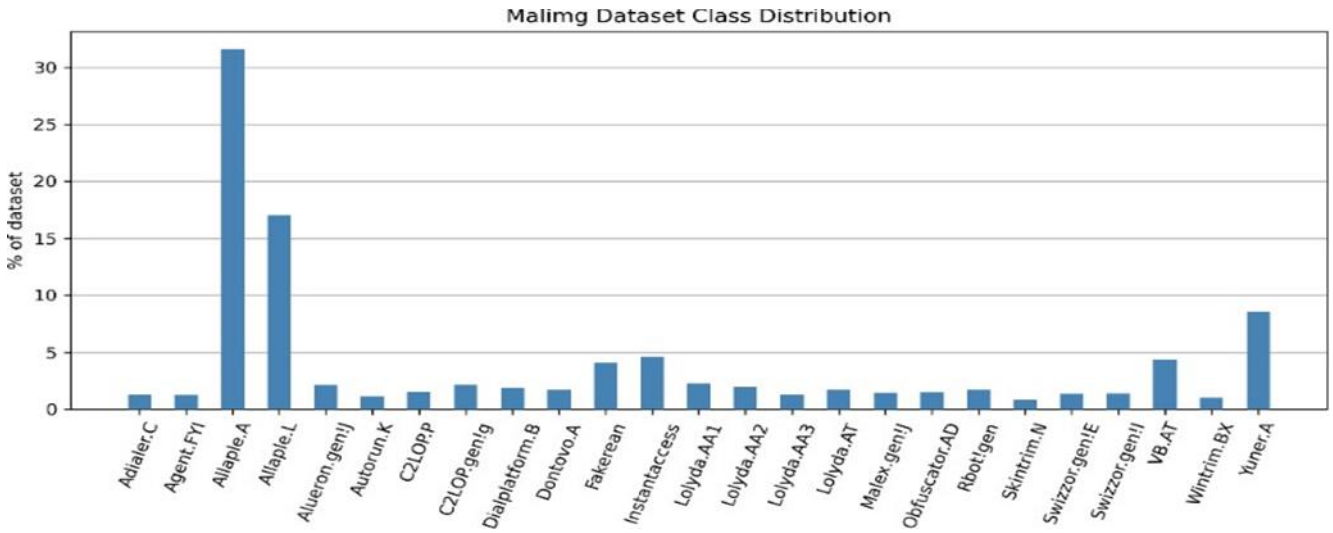

Fig 1: Malimg Class distribution

Malimg, a Windows Pervasive Executable (PE) file were transformed into images. The components of a PE file are shown in Figure 2. The executable code is located in the '.text' file is the default code section. '. rdata ' is the read-only file format by default. String literals and C++/COM lables are found here. Next '.data' files are the standard read/write data files. This is the common location for storing global variables. Then '.rsrc' is the section stores the module's resources and any icons that the program makes use of. (Microsoft, n.d.)

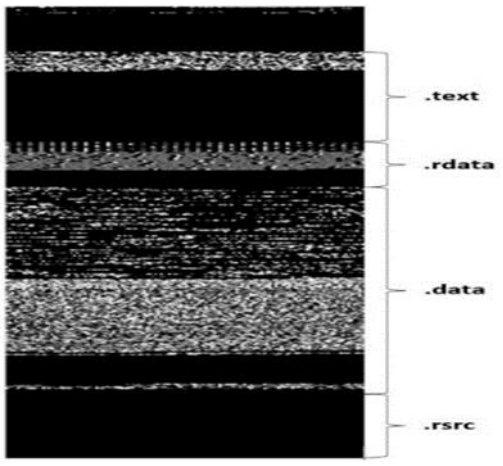

Fig 2: Trojan Code Part

Malware binaries are a sequence of 8-bit unsigned integers, which are arranged into a two-dimensional array. This may be represented graphically as a [0, 255] grayscale picture (where 0 is black and 255 is white). The height of the image may change with the file size, while the width remains constant [13]. Multiple completely black regions may be seen is the text part in Figure 2. This might be uninitialized code, but it's more likely to be the result of zero padding, a frequent kind of code obfuscation. Implementing a static approach has the advantage that CNN is less likely to be affected by code obfuscation strategies. To create the subsequent visual data sample by converting all images in the dataset into 64x64 the grayscale images is to visualize the highly compressed image.

IV. Results and Discussion

Total Training Time and Average Training Time across Epochs were analyzed for comparing the five CNNs evaluated in this experiment for performance as Complete Accuracy, Loss, F1 Score, Precision and Recall. To determine which network performed best and which would be the most suited to operate on a low-powered device in terms of accuracy and overall strength of an Internet of Things (IoT) device. The minimal amount of processing power is needed to determine a file is harmful or not is the strength of static malware analysis. Experiments were conducted with a Multi-Layer Convolutional Neural Network (MLCNN) for classifying the malware with high efficiency. A training/validation split of all the models are 70%/30%. Adam on their layers and categorical cross entropy as the loss function are used.

4.1 Feature Extraction using GLRLM

Nowadays, a plethora of texture analysis techniques have emerged, leukocyte nuclei are characterized by 11 textural traits are determined using grey level run length matrices. The number of grey levels (g) and the longest run (r) (a series of consecutive pixels with the same grey level strength in a single linear direction) are both quantified for a sub-image of dimension M x N. The sum of all runs is symbolized by the letter Z. The GLRLM is a two-dimensional (g, r) matrix with every component q (m, n) indicating the frequency with the run of length n and grey level m occurs along the specified axis.

4.2 Convolution Models Architecture

Five different CNN models are used for classifications. Batch normalization is used in Models 1 and 2, whereas Local Response Normalization is used in Models 3, 4, and 5. Two models are compared with the same 64x64x1 input size but different numbers of MLCNN layers to shorten

the intervals between training epochs. Models 3, 4, and 5 normalization techniques scale well with varying input sizes. Batch normalization tested on malware binaries proved to be ineffective, byte values from ASCII text, code, structured-data, images, etc have different meaning. The contextual sensitive information of a binary file, may be vulnerable by the cropping and shifting of the batch data during normalization. The feature's original context is lost due to the scaling and shifting of the image throughout the batch standardization process. If the image is scaled and shifted such that a malicious string representing an API call in a Windows PE file is moved to another region of the PE file structure, the learning algorithm will no longer understand the significance of the original sample.

Instead of scaling and shifting, the principle of lateral inhibition is applied. This lateral inhibition is local contrast enhancement for local response normalization, the highest pixel values is the excitation for the next subsequent layers [7]. In contrast to batch normalization, which shifts and scales an image picture based on its gamma and beta values, local response normalization takes into account the immediate surroundings of a single pixel. In light of the binary data's inherent context, the local response normalization function was deemed superior to the batch normalization approach. Local response normalization was applied for the remaining models after batch normalization was tried in models 1 and 2 of input size 64x64x1 with 25 epochs. After confirming its efficacy via experimentation to try out various input sizes for both our training and testing data. This is also why models 3, 4, and 5 were designed to work well with input data of varying sizes as 128x128x1(10 epochs), 64x64x1 (10 epochs) and 32x32x1(50 epochs).

Model	Layers	Methods	Time in minutes	Accuracy	Loss	F1-Score	Precision	Recall
1	30 &15 filters, Relu, 3x3 kernel	Batch Normalization	9.83	83.12%	30.11%	0.82	0.83	0.83
2	32, 64 &128 filters, Relu, 3x3 kernel	Batch Normalization	5.97	86.62%	15.12%	0.86	0.86	0.87
3	50 & 70 filters, Relu, 3x3 kernel	Batch Normalization	4.05 hours	96.25%	27.88%	0.95	0.95	0.96
4	25 & 35 filters, Relu, 3x3 kernel	Local Response Normalization	18.20	91.78%	22.22%	0.91	0.91	0.92
5	15 & 25 filters, Relu, 3x3 kernel	Local Response Normalization	11.98	75.99%	129.89%	0.75	0.75	0.76

Table 2: Performances of the CNN Models

Models 1 and 2 employ a 64x64x1 input size, 25 epochs of training, and a batch normalization algorithm. Model 1 performed poorly compared to the other models, training time was less than 10 minutes makes growth in the accuracy. Model 2 created to speed up learning, improve precision and achieve an increasing accuracy. Figure 3 shows the accuracy curves during training and validation for Model 2. Model 3, 128x128x1 input size, achieved the highest recorded accuracy by increasing the input shape from the previous two models' 64x64x1 input form. Due

to the excessively long training period required by model 3, 64x64x1 input structure for model 4 showed promising performance outcomes after undergoing the same amount of training epochs as Model 3. In model 5, 32x32x1 has the performance reduced during training and validation for both models. Figure 4 shows the accuracy curves during training and validation for model 4. Figure 5 shows during all 10 epochs of model 4 accuracy curve. Figure 5 shows the model 5 the validation accuracy curve deviates significantly from the training curve.

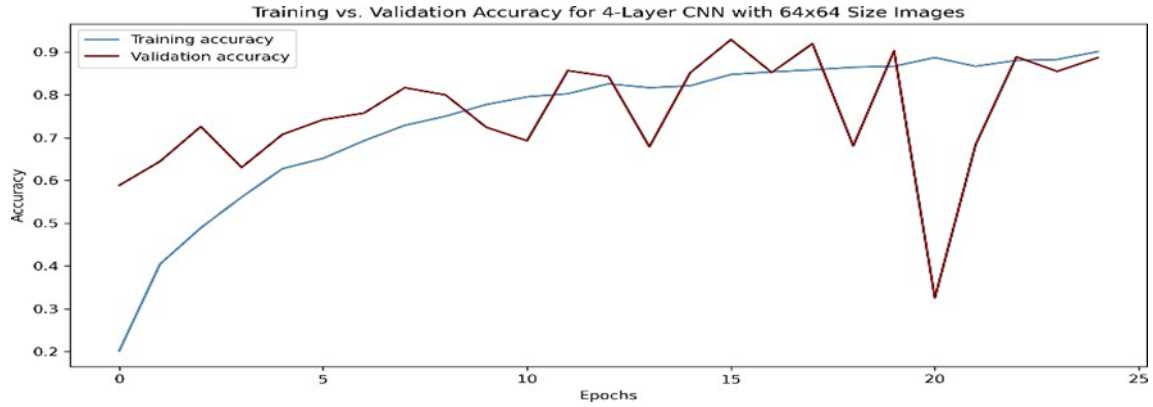

Fig 3: Model 2's Training and Validation Accuracy

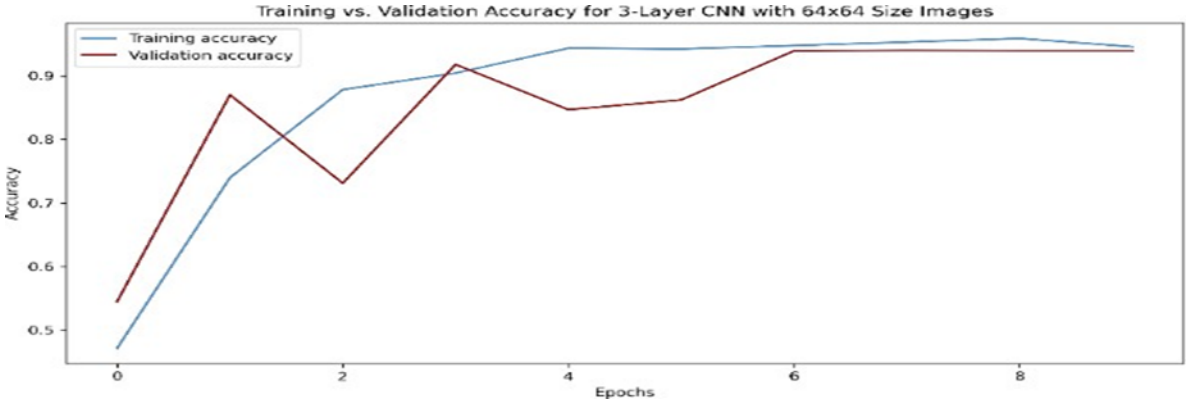

Fig 4: Model 4's Training and Validation Accuracy

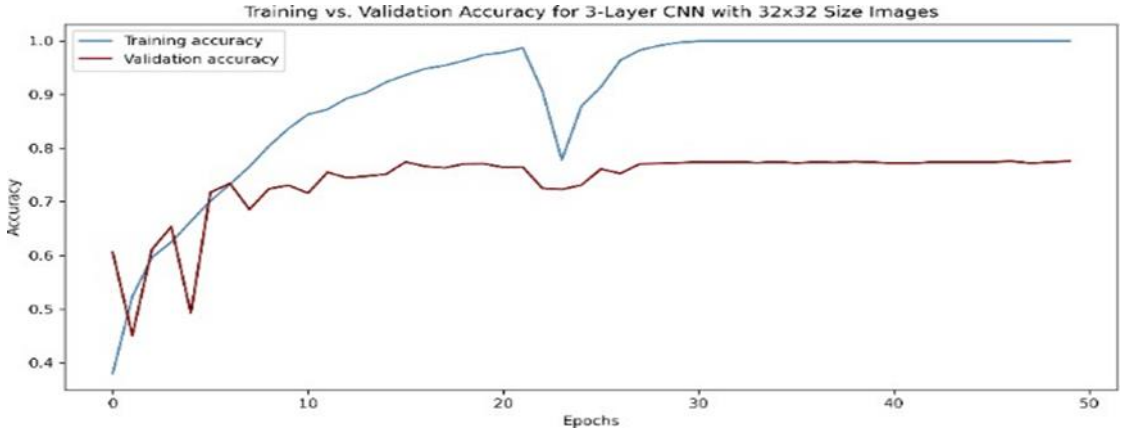

Fig 5: Model 5's Training and Validation Accuracy

IV. Conclusion

Real time raw PE file usage in malware extractions needs to face some difficulties like run in virtual environment, quarantine malicious data, gray scale conversion, etc for the researchers. Malware research will remain behind other areas of machine learning unless malware investigators and analysts have an open and freely available dataset to do experiments with. Since the PNG images of Malimg dataset were transformed to true monochrome using Python's image module with GPU acceleration. Multi-Layer Convolutional neural networks and Gray Level Run Length Matrix training require a lot of processing power. Static malware detections are the first stage of defense against harmful attacks would be the

best suggested malware detection model system. Machine learning, in contrast to natural language processing, is not a solution for all cyber security problems. Malware researchers would have more time to obtain more comprehensive knowledge regarding the dangerous file at the first phases of static malware analysis. Based on the findings, the static malware approach described has the potential of detecting malware in real-time.

References

[1] Muhammad Shoaib Akhtar (2022) Detection of Malware by Deep Learning as CNN-LSTM Machine Learning Techniques in Real Time.

[2] Rong Wang (2021) Malware Detection using CNN via word embedding in cloud computing infrastructure.

[3] Kaspersky Lab (2021). The number of new malicious files detected every day increases by 5.2% to 360,000 in 2020. [online] www.kaspersky.com. <Available at: https://www.kaspersky.com/about/press-releases/2020_the-number-of-new- malicious-files-detected-every-day-increases-by-52-to-360000-in-2020> .

[4] ISC2 (2021). 2021 Cybersecurity Workforce Study. [online] www.isc2.org, ISC2, pp.24–25. <Available at: https://www.isc2.org//-/media/ISC2/Research/2021/ISC2-Cybersecurity-Workforce-Study-2021.ashx>.

[5] SonicWall (2021, 2023). SonicWall Cyber Threat Report. [online] https://www.sonicwall.com/, Milpitas, CA: SonicWall Inc.,p.58. <Available at: https://www.sonicwall.com/resources/white-papers/2021-sonicwall-cyber-threat-report/ >.

[6] Krithika V. (2021). Malware and Benign Detection Using Convolutional Neural Network <Available at : DOI:10.1007/978-981-16-0171-2_4>.

[7] Anwar A. (2021). Difference between Local Response Normalization and Batch Normalization. [online] Medium. <Available at: http://towardsdatascience.com/difference-between-local-response-normalization-and-batch-normalization- 272308c034ac> .

[8] Mallet H. (2020). Malware Classification using Convolutional Neural Networks — Step by Step Tutorial. [online] Medium. <Available at: https://towardsdatascience.com/malware-classification-using-convolutional-neural-networks-step-by- step-tutorial-a3e8d97122f>.

[9] Véstias M.P. (2019). A Survey of Convolutional Neural Networks on Edge with Reconfigurable Computing. Algorithms, 12(8), p.154.

[10] Bhodia N., Prajapati P., Troia F. and Stamp M. (2019). Transfer Learning for Image-Based Malware Classification.

[11] Le Q., Boydell O., Mac Namee B. and Scanlon M. (2018). Deep learning at the shallow end: Malware classification for non-domain experts. Digital Investigation, [online] 26, pp.S118–S126. <Available at: https://www.sciencedirect.com/science/article/pii/S1742287618302032>.

[12] Gibert D., Matteu C., Planes J. and Vicens R. (2018). Using convolutional neural networks for classification of malware represented as images. Journal of Computer Virology and Hacking Techniques, 15(1), pp.15–28.

[13] Nataraj L., Karthikeyan S., Jacob, G. and Manjunath B.S. (2011). Malware images. Proceedings of the 8th International Symposium on Visualization for Cyber Security - VizSec ’11.

[14] Raff E., Barker J., Sylvester J., Brandon R., Catanzaro B. and Nicholas C. (2017). Malware Detection by Eating a Whole EXE.

[15] Microsoft (n.d.). An In-Depth Look into the Win32 Portable Executable File Format, Part 2: Figures. [online] bytepointer.com. <Available at: https://bytepointer.com/resources/pietrek_in_depth_look_into_pe_format_pt2_figures.htm>.