Grenze International Journal of Engineering and Technology, June Issue

Enhancing Cyber Security Defences through Accurate Malware Classification

K.R.Nandhashree[1], Nijin P[2], Selva Balachandar M[3] and Sivaraman S[4]
[1]Assistant Professor, Department of Cyber Security, SRM Valliammai Engineering College
Email: Nandhashree29@gmail.com
[2-4]UG Scholar, Department of Cyber Security, SRM Valliammai Engineering College
Email: paulvijinijin5555@gmail.com, selvabala21m@gmail.com, aadhisiva65@gmail.com

***Abstract*— The rapid integration of virtual environments into human lives, accelerated by the COVID-19 pandemic, has shifted criminal activities to the digital realm, where malware serves as a primary tool for cybercriminals. Conventional detection methods struggle against the evolving sophistication of malware. To address this challenge, a novel approach leveraging deep learning through the Customized Deep Learning-based Malware Classification (CDL-MC) architecture is proposed for detecting new and complex malware types. Objectives include developing an image-based malware dataset and implementing CDL-MC with multiple convolutional and fully connected layers. Additionally, the model is deployed in a user-friendly MATLAB GUI application to enhance practical usability in cybersecurity. Experimental results demonstrate the efficacy of the method in classifying malware with high accuracy, surpassing existing state-of-the-art methods. By harnessing deep learning, the approach offers a promising solution to combat ever-evolving malware threats and ensure enhanced cybersecurity in the virtual age.**

***Index Terms*— Malware analysis, CDL-MC, Deep Learning, Image Dataset.**

I. INTRODUCTION

In today's digitally interconnected world, the proliferation of malware poses an ever-present threat to individuals, businesses, and organizations worldwide. Malicious software, ranging from viruses and worms to ransomware and advanced persistent threats (APTs), continues to evolve in sophistication and complexity. The consequences of malware attacks can be severe, resulting in data breaches, financial losses, and disruption of critical services. As cybercriminals exploit vulnerabilities in systems and networks with growing expertise, the need for robust and proactive malware analysis approaches becomes paramount. More traditional methods are implemented. However, in the malware analysis, several challenges persist, hampering the efficacy of existing detection and mitigation strategies. Conventional approaches often rely on signature- based detection methods, which struggle to keep pace with the rapid evolution of malware variants and the proliferation of polymorphic and metamorphic techniques used by cybercriminals to evade detection. Moreover, the sheer volume and diversity of malware samples pose significant scalability and efficiency challenges for analysts, who must sift through vast datasets to identify new threats. Additionally, the lack of explainability and interpretability in some machine learning-based approaches limits their utility in providing actionable insights into malware behavior and characteristics.

To address these pressing challenges, this paper proposes a novel approach to malware analysis that leverages

Grenze ID: 01.GIJET.10.2.204

advanced deep learning techniques, specifically tailored for efficient detection and classification of diverse malware types. By harnessing the power of deep neural networks, the proposed methodology aims to enhance the accuracy, scalability, and interpretability of malware analysis, thereby empowering cybersecurity professionals to stay ahead of evolving cyber threats. Through rigorous experimentation and validation, this work seeks to establish the effectiveness and practical applicability of the proposed approach in bolstering cybersecurity defenses against the ever-evolving landscape of malware threats.

The paper comprises several key sections. Section II provides an overview of related works in the field of malware analysis, discussing existing methodologies and their limitations. In Section III, the proposed methodologies are explained in detail, outlining the Customized Deep Learning-based Malware Classification (CDL-MC) architecture and its innovative features. Section IV presents the results of experiments conducted using the proposed approach, followed by a comprehensive discussion of the findings. Finally, Section V concludes the paper by summarizing the contributions, highlighting the significance of the research, and suggesting avenues for future work in the field of malware analysis.

II. Related Works

Malware analysis is a critical process aimed at understanding the nature and behaviors of malicious software [1]. This analysis is indispensable as it not only facilitates malware detection but also provides insights into various aspects of the malware's characteristics. During the analysis, several key questions can be addressed: the structure of the malware can be visualized, the methods of infection and spread can be uncovered, and the extent of damage inflicted on victim machines can be evaluated [1]. Malware analysis typically falls into two main categories: static and dynamic. It begins with basic static analysis and progresses to more advanced dynamic analysis [2]. Analysis can be conducted manually or automatically, with manual analysis requiring domain expertise, while automatic analysis necessitates advanced programming skills in data science.

Static analysis involves identifying the structure of a malware sample without executing its code [3]. It comprises two main parts: basic and advanced static analysis. In basic static analysis, file strings, header information, and functions are examined superficially, providing an initial understanding of the program's characteristics [1]. Tools like PEiD, BinText, MD5deep, and PEview are commonly used to extract this information [2]. Basic static analysis serves as the initial step in malware analysis, laying the groundwork for further exploration. To delve deeper into the malware's inner workings, advanced static analysis is conducted. Here, the program commands are scrutinized in detail, often using disassemblers to generate assembly codes from machine codes [4]. Tools such as IDA Pro and its Hex-Rays decompiler extension are widely employed for advanced static analysis. During this process, assembly instructions are meticulously examined to uncover the malware's characteristics and functionalities. Advanced static analysis yields profound insights into the malware's purpose and behavior. However, conducting this analysis requires a high level of expertise in assembly code instructions and operating system concepts [1], [5].

Dynamic analysis involves executing malware instructions to assess its behaviors, commonly conducted in closed environments like sandboxes or virtual machines [4]. During this process, various aspects such as function calls, information flows, and network activities are observed. It provides a more accurate understanding of malware functionality compared to static analysis. Dynamic analysis is categorized into basic and advanced, with basic methods utilizing monitoring tools like Process Monitor and Wireshark, while advanced methods employ debuggers such as OllyDbg and WinDbg, allowing for in-depth examination of malware functionalities [1]. However, advanced dynamic analysis requires expertise in assembly-level instructions and operating system concepts.

Griffin et al. [6] introduced an automatic signature extraction method that utilizes various library identification techniques and diversity-based heuristics to extract string signatures from malware files, resulting in a reduction in false identifications. Tang et al. [7] presented a simplified regular expression signature using bioinformatics techniques for detecting polymorphic worms, involving three phases: identifying prominent sub-sequences, eliminating noise, and ensuring compatibility with existing Intrusion Detection Systems (IDSs). Liu and Sandhu [8] proposed a fingerprint-based signature generation method for hardware-based malware detection, employing cryptographic hash-based signatures to detect Trojans embedded in hardware.

Kolbitsch et al. [9] introduced a malware detection system based on a graph model, where system calls are transformed into a diagram representing transitions among system calls. The program diagram is compared with existing diagrams to identify malware or benign files, with new behaviors dynamically added during analysis. Lanzi et al. [10] proposed a system-centric behavioral model, utilizing differences in how malicious and benign software interact with system resources to generate behavior sequences from system calls. Chandramohan et al.

[11] presented the Behavior- Oriented Feature Model (BOFM) for malware detection, converting interrelated system calls into meaningful behaviors to create feature vectors for classification using machine learning algorithms. Singh et al. [12] utilized behavior-based multiple API system calls for malware detection, creating multiple API sequences and classifying them using machine learning algorithms. Aslan et al. [13] proposed a behavioral-based SCBM model, capturing semantically related features from analyzed program samples to differentiate malicious behavior patterns from benign, achieving efficient detection of known and unknown malware.

Finally, a novel hybrid approach incorporating dynamic analysis, cyber threat intelligence, machine learning, and data forensics is presented in [14], aiming to predict IP reputation and categorize zero-day attacks using behavioral analysis, with satisfactory performance compared to existing methods. Similarly, a behavioral-based detection method called APTMalInsight is proposed in [15], highlighting the need for innovative solutions in the evolving landscape of malware detection, aligning with the objectives of our proposed work.

III. PROPOSED METHODOLOGY

The proposed system comprises several key components aimed at enhancing malware detection capabilities, as illustrated in Figure 1. Initially, an image-based malware dataset is constructed using MD5 hash values, with each binary vector converted into a matrix representing pixel values. This dataset serves as input for the Customized Deep Learning based Malware Classification (CDL-MC) architecture, which incorporates multiple convolutional and fully connected layers for efficient feature extraction and classification of malware samples. The training process includes diverse image augmentations to enhance dataset variability and hyperparameter optimization to ensure robust model training. Subsequently, the model undergoes rigorous testing on unseen data, evaluating performance using accuracy, precision, recall, specificity, and F1-score metrics. Finally, the trained model is deployed in a user-friendly MATLAB GUI application, providing an accessible interface for real-world cybersecurity applications. This end-to-end system offers a comprehensive solution for effective and efficient malware detection, leveraging advanced deep learning techniques and user-friendly interface design.

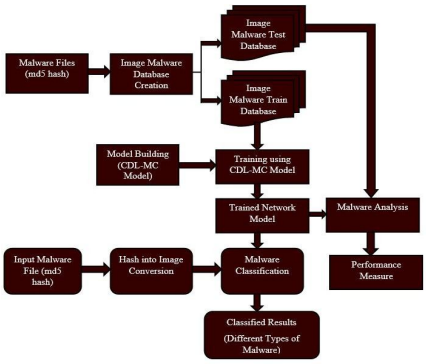

Figure 1. Overall proposed methodology for malware image analysis.

A. Image Malware Dataset Creation

To begin, gather the malware dataset identified by their MD5 hash values, containing binary files representing various malware samples. For each MD5 hash value, convert it into an 8-bit binary vector, as MD5 hashes are typically 128-bit long, allowing each of the 16 bytes to be represented as an 8-bit binary value. Next, convert the 8-bit binary vector into its corresponding decimal value. Organize the resulting decimal values into a matrix, where each row represents a malware sample. Interpret each row of the matrix as pixel values for an image. Depending on the desired representation, reshape the matrix into a 2D array considering width and height, and visualize or save these arrays as images, which is shown in Figure 2.

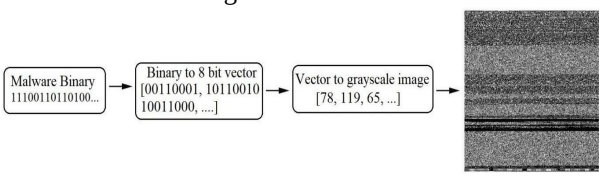

Figure 2. Image malware creation.

B. CDL-MC Model

After creating the image dataset, our focus shifted to the development of the CDL-MC architecture, a pivotal step in feature extraction and malware classification. The CDL-MC model we devised comprises several weight layers, each tuned to learn intricate patterns from the input data, which is standardized to 64 by 64 pixels. Structurally, the CDL-MC architecture encompasses multiple convolutional layers, enabling the extraction of complex features. These layers are strategically interleaved with max pooling operations, enhancing the model's ability to capture salient information while reducing dimensionality. Furthermore, our model incorporates two fully connected layers. These densely connected layers serve as critical nodes for high-level feature synthesis, allowing the model to discern intricate relationships within the extracted features.

(i) Structure of CDL-MC Model

The CDL-MC model is composed of nine weight layers, featuring six convolutional layers and three fully connected layers. The structure of the CDL-MC model is shown in Figure 3. These layers are strategically designed to address issues of overfitting and underfitting while optimizing computational efficiency.

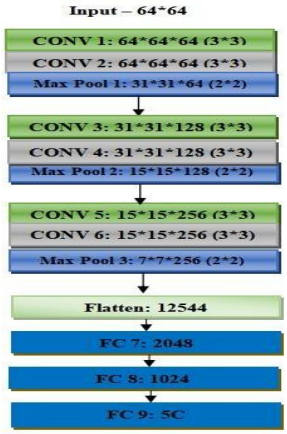

Figure 3. CDL-MC model structure.

i. The first two layers consist of convolutional layers with 3x3 filters, each employing 64 filters. This configuration results in a volume of 64x64x64 for the first and second convolutional layers.
ii. Subsequently, a pooling layer with a max-pool of 2x2 size and a stride of 2 is applied to reduce the volume's height and breadth from 64x64x64 to 31x31x64.
iii. The third and fourth convolutional layers utilize 128 filters, resulting in a new dimension of 31x31x128.
iv. Upon employing the pooling layer, the volume is decreased to 15x15x128.
v. Additional convolutional layers are added, featuring 256 filters each, resulting in a volume size of 15x15x256 for the fifth and sixth convolutional layers. Subsequently, downsampling reduces the size to 7x7x256.
vi. The resulting 7x7x256 volume is flattened into a Fully Connected (FC) layer with 2048 and 1024 neurons, followed by a softmax output of 5 classes after the FC layers.

C. Malware Image Analysis

After constructing the CDL-MC model, our next step involved loading the training set image dataset. To enhance the robustness and variability of the training data, we applied diverse image augmentation techniques, including reflection, translation, scaling, and flipping. These augmentations expanded the dataset, ensuring the model's exposure to a wide array of potential inputs. The training process commenced, focusing on the optimization of hyperparameters across epochs. Iteratively, fine-tuned crucial factors such as learning rates and batch sizes, leveraging the SGDM optimizer. This meticulous calibration was fundamental in crafting a resilient and well-trained model. Through this process, the CDL-MC model gained the capability to discern intricate patterns and generalize effectively, reinforcing its proficiency in malware classification.

Following the training process, the CDL- MC model underwent rigorous testing to assess its performance on unseen data. A separate set of malware samples, distinct from the training dataset, was used to evaluate the model's ability to generalize and accurately classify diverse malware types. The testing phase involved inputting new malware images into the trained model and analyzing its predictions. This step aimed to ensure that the CDL-MC architecture effectively translates its learned features into accurate classifications for real-world scenarios.

D. GUI Integration

In the MATLAB GUI, users input an MD5 hash file, which is then processed to generate an image representation of the malware. This image undergoes classification using a trained model, allowing the GUI to identify and categorize the malware into different types based on learned patterns and features. Through this seamless process, users can efficiently analyze and classify malware samples, aiding in cybersecurity efforts and threat detection.

IV. Experimental Results

In this section, the performance of the CDL-MC model was rigorously evaluated using a diverse range of metrics, encompassing accuracy, precision, recall, specificity, and F1-score. These metrics were meticulously calculated by comparing the model's predictions against the ground truth labels of the testing dataset, providing a comprehensive assessment of its efficacy. This visualization facilitated in-depth analysis of the model's classification outcomes across various malware types, shedding light on specific classification patterns and potential challenges encountered during the classification process. Through this thorough evaluation, the strengths and areas for improvement of the CDL-MC model were elucidated, informing future refinements and optimizations aimed at enhancing its performance in real-world cybersecurity applications.

$$ccuacy = \frac{Pa+Na}{Pa+Na+Pa+Na} \quad (1)$$

$$Pcso = \frac{(Pa)}{(Pa+Pa)} \quad (2)$$

$$ca = \frac{(Pa)}{(Pa+Na)} \quad (3)$$

$$pccty = \frac{(Na)}{(Na+Pa)} \quad (4)$$

$$= 2 \times \frac{Pcso \times ca}{Pcso+ca} \quad (5)$$

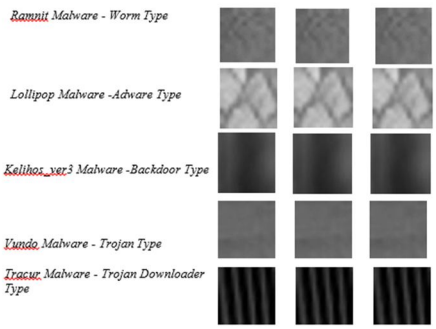

Figure 4. Created malware dataset sample images.

The results of the experiment showcase a comprehensive analysis of the CDL-MC model's performance in malware classification. Initially, an image dataset was created from hash values, and the process was illustrated in Figure 4. The dataset comprised a diverse range of malware types, allowing for thorough testing of the model's capabilities.

Throughout the training process, detailed insights into the learning dynamics and convergence of the model were obtained through the plotting of accuracy and loss graphs for the malware image dataset. Figure 5 illustrates these graphs, which served as essential diagnostic tools for monitoring the model's performance over successive epochs. The accuracy graph depicted the model's ability to correctly classify malware samples over time, with increasing accuracy indicative of improved learning and model refinement. Concurrently, the loss graph demonstrated the reduction in the model's loss function, reflecting the minimization of errors during training. By visually tracking these metrics, patterns of improvement and convergence emerged, providing critical feedback for fine-tuning model parameters and optimizing training strategies. Overall, the accuracy and loss graphs played a pivotal role in

gauging the training progress and validating the effectiveness of the training process in enhancing the model's classification capabilities for malware detection.

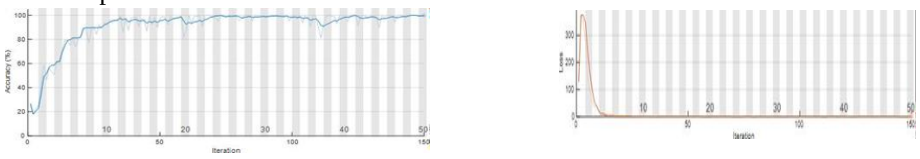

Figure 5. Training accuracy and loss.

During the model testing phase, the system seamlessly executed the conversion of hash values into images and conducted a comprehensive classification of malware types and their associated impact. As depicted in Figure 6, this step exemplified the practical utility of the CDL-MC model in real- world scenarios, showcasing its adeptness at accurately identifying malware types based on input hashes. By leveraging the learned features and classification capabilities of the model, the system effectively distinguished between various malware variants and provided valuable insights into their potential impact on system security and integrity. This successful demonstration underscored the robustness and efficacy of the CDL-MC architecture in facilitating precise and reliable malware classification, thereby validating its suitability for practical cybersecurity applications.

From Figure 6, the classified malware type for sample 1 is identified as Tracur Malware— Trojan-Downloader Type, with its impact illustrated in same figure.

The work concludes with a comprehensive performance analysis, featuring a confusion matrix that vividly illustrates the classification outcomes for five distinct malware classes: Ramnit Malware (Worm Type), Lollipop Malware (Adware Type), Kelihos_ver3 Malware (Backdoor Type), Vundo Malware (Trojan Type), and Tracur Malware (Trojan Downloader Type), as depicted in Figure 7. This analysis provides valuable insights into the CDL-MC model's effectiveness in categorizing diverse malware types, thereby enhancing cybersecurity defenses.

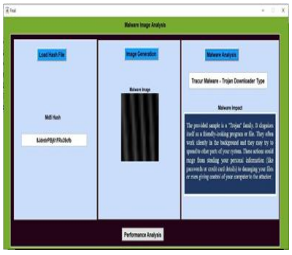

Figure 6. Testing phase results for input sample 1.

Figure 7. Confusion matrix of CDL-MC for different types of malware analysis.

Additionally, key performance metrics, including accuracy, precision, recall, specificity, and F1-score, were computed and are summarized in Table 1. The analysis reveals an accuracy of 97.96%, precision of 97.96%, recall of 97.96%, specificity of 99.49%, and an F1-score of 97.96% for the classification of malware types. These metrics underscore the robustness and efficacy of the CDL- MC model in accurately identifying and classifying diverse malware variants, thus fortifying cybersecurity measures.

Similarly, the performance analysis graph visualization is presented in Figure 8, providing a comprehensive overview of the model's classification performance across different malware types.

TABLE I. CDL-MC MODEL PERFORMANCE ANALYSIS

Metrics	Proposed
Accuracy	97.96%
Precision	97.96%
Recall	97.96%
Specificity	99.49%
F1-Score	97.96%

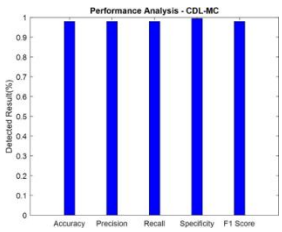

Figure 8. Performance analysis of CDL-MC model.

V. CONCLUSION

In cybersecurity, the perpetual evolution of malware poses an enduring challenge, exacerbated by sophisticated code obfuscation and packing techniques. Addressing this challenge, our work presents the CDL-MC architecture,

a novel solution tailored for effective malware detection. A pivotal contribution lies in the introduction of an image- based malware dataset, enriching the comprehension of malware behaviors. Leveraging this dataset, the CDL-MC architecture harnesses multiple convolutional layers to extract nuanced features from image-based malware samples, enabling precise classification. Rigorous experimentation confirms the efficacy of our approach, with the CDL-MC architecture achieving remarkable accuracy, precision, recall, specificity, and F1-score rates, notably reaching 98.00%. By adeptly classifying malware types, our method empowers cybersecurity systems to proactively detect and neutralize diverse malware variants, enhancing digital resilience. This work not only fortifies current cybersecurity measures but also lays the groundwork for future advancements in malware detection methodologies, fostering a safer digital landscape.

REFERENCES

[1] M. Sikorski and A. Honig, Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software. San Francisco, CA, USA: No starch press, 2012.

[2] Ö. Aslan, ‘‘Performance comparison of static malware analysis tools versus antivirus scanners to detect malware,’’ in Proc. Int. Multidisciplinary Stud. Congr. (IMSC), 2017, pp. 1–6.

[3] S. K. Pandey and B. M. Mehtre, ‘‘Performance of malware detection tools: A comparison,’’ in Proc. IEEE Int. Conf. Adv. Commun., Control Comput. Technol., May 2014, pp. 1811–1817.

[4] Ö. Aslan and R. Samet, ‘‘Investigation of possibilities to detect malware using existing tools,’’ in Proc. IEEE/ACS 14th Int. Conf. Comput. Syst. Appl. (AICCSA), Oct. 2017, pp. 1277–1284.

[5] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, ‘‘Robust intelligent malware detection using deep learning,’’ IEEE Access, vol. 7, pp. 46717–46738, 2019.

[6] K. Griffin, S. Schneider, X. Hu, and T. C. Chiueh,‘‘Automatic generation of string signatures for malware detection,’’ in Proc. Int. Workshop Recent Adv. Intrusion Detection. Berlin, Germany: Springer, 2009, pp. 101–120.

[7] Y. Tang, B. Xiao, and X. Lu, ‘‘Using a bioinformatics approach to generate accurate exploit- based signatures for polymorphic worms,’’ Comput. Secur., vol. 28, no. 8, pp. 827–842, Nov. 2009.

[8] B. Liu and R. Sandhu, ‘‘Fingerprint-based detection and diagnosis of malicious programs in hardware,’’ IEEE Trans. Rel., vol. 64, no. 3, pp. 1068–1077, Sep. 2015.

[9] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X. Y. Zhou, and X. Wang, ‘‘Effective and efficient malware detection at the end host,’’ in Proc. USENIX Secur. Symp., vol. 4, 2009, pp. 351–366.

[10] A. Lanzi, D. Balzarotti, C. Kruegel, M. Christodorescu, and E. Kirda, ‘‘AccessMiner: Using system-centric models for malware protection,’’ in Proc. 17th ACM Conf. Comput. Commun. Secur. (CCS), 2010, pp. 399–412.

[11] M. Chandramohan, H. B. K. Tan, L. C. Briand,mL. K. Shar, and B. M. Padmanabhuni, ‘‘A scalable approach for malware detection through bounded feature space behavior modeling,’’ in Proc. 28th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE), Nov. 2013, pp. 312–322.

[12] A. Singh, R. Arora, and H. Pareek, ‘‘Malware analysis using multiple API sequence mining control flow graph,’’ 2017, arXiv:1707.02691. [Online]. Available: http://arxiv.org/abs/1707.02691

[13] Ö. Aslan, R. Samet, and Ö. Ö. Tanrıöver, ‘‘Using a subtractive center behavioral model to detect malware,’’ Secur. Commun. Netw., vol. 2020, pp. 1–17, Feb. 2020.

[14] N. Usman, S. Usman, F. Khan, M. A. Jan, A. Sajid, M. Alazab, and P. Watters, ‘‘Intelligent dynamic malware detection using machine learning in IP reputation for forensics data analytics,’’ Future Gener. Comput. Syst., vol. 118, pp. 124–141, May 2021.

[15] W. Han, J. Xue, Y. Wang, F. Zhang, and X. Gao, ‘‘APTMalInsight: Identify and cognize APT malware based on system call information and ontology knowledge framework,’’ Inf. Sci., vol. 546, pp. 633– 664, Feb. 2021.

[16] Et. al., Selvarathi C,. “A Survey on Machine Learning Approach to Detect Malware.” (2021).

[17] Roseline, S. Abijah et al. “Intelligent Vision- Based Malware Detection and Classification Using Deep Random Forest Paradigm.” IEEE Access 8 (2020): 206303-206324.

[18] Atitallah, Safa Ben et al. “A Novel Detection and Multi-Classification Approach for IoT-Malware Using Random Forest Voting of Fine-Tuning Convolutional Neural Networks.” Sensors (Basel, Switzerland) 22 (2022): n. pag.

[19] Nisa, Maryam et al. “Hybrid Malware Classification Method Using Segmentation-Based Fractal Texture Analysis and Deep Convolution Neural Network Features.” Applied Sciences (2020): n. pag.

[20] Chong, Xulei et al. “Classification of Malware Families Based on Efficient-Net and 1D-CNN Fusion.” Electronics (2022): n. pag.

[21] Ö. Aslan and R. Samet, ‘‘A comprehensive review on malware detection approaches,’’ IEEE Access, vol. 8, pp. 6249–6271, Jan. 2020.

[22] R. Komatwar and M. Kokare, ‘‘A survey on malware detection and classification,’’ J. Appl. Secur. Res., pp. 1–31, Aug. 2020.

[23] A. A. Yilmaz, M. S. Guzel, E. Bostanci, and I. Askerzade, ‘‘A novel action recognition framework based on deep-learning and genetic algorithms,’’ IEEE Access, vol. 8, pp. 100631–100644, 2020.