2024 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET) | 979-8-3503-8992-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICRAMET62801.2024.10809101

Image Classification-Based Ransomware Attack Detection Using Deep Learning Algorithm

Shafii Muhammad Abdulhamid
Department of Information Technology
Community College of Qatar
Lusail, Qatar
shafii.abdulhamid@ccq.edu.qa

Fawaz Issa Mohammed Albalushi
Department of Information Technology
Community College of Qatar
Lusail, Qatar
fawaz81041@ccq.edu.qa

Mohammed Hamed Al-Kuwari
Department of Information Technology
Community College of Qatar
Lusail, Qatar
mohammed81246@ccq.edu.qa

Yousef Abdullah Al-Sulaiti
Department of Information Technology
Community College of Qatar
Lusail, Qatar
yousuf62109@ccq.edu.qa

Nadim Rana
Department of Computer Science
Jazan University
Jazan, Saudi Arabia
nadimrana@jazanu.edu.sa

Abdullah Hussein Al-Ghushami
Department of Information Technology
Community College of Qatar
Lusail, Qatar
abdullah.alghushami@ccq.edu.qa

***Abstract*—Ransomware represents a significant cybersecurity threat that can potentially damage corporate entities. Conventional antivirus methodologies must be revised to mitigate emerging and complex cyber-attack vectors. This study introduces an architecture for a Convolutional Neural Network (CNN) predicated on deep learning paradigms; it utilizes a dataset composed of both malicious and benign executables. These files were transformed into grayscale images via the Python Imaging Library (PIL) and subsequently processed by the CNN to effectively discriminate between images indicative of ransomware and those classified as benign. The effectiveness of the CNN model was measured using metrics like accuracy, precision, recall, and the F1 score. The training accuracy was 0.9476, while the validation accuracy was 0.9383 percent. Moreover, the precision, F1 score, and recall of training were higher than their respective validation rates. This model can contribute to developing an effective solution for detecting ransomware attacks, complement existing approaches, and improve accuracy.**

***Keywords—Ransomware, Cybersecurity, Deep learning algorithm, Convolutional Neural Network, Recurrent Neural Network*s**

I. Introduction

Ransomware is malicious software designed to damage or exploit computer applications, systems, and networks. Programmers can create and customize it to perform functions based on the attacker's plan. Moreover, it works differently than other viruses because once it is infected, it can prevent the user from accessing data by encrypting the files with asymmetric cryptography based on two pairs of keys: the public key, which encrypts the data and files, and the private key, which remains with the attacker [1, 2].

There are several methods of spreading ransomware, including phishing emails, which attackers most commonly use to trick the recipient receiver into opening a link that directs the victim to a fake website or downloading an attached file that contains malicious code that can infect the victim's computer and demands a ransom amount to interchange the decryption key. Even if the ransom is paid, the data or files will not be guaranteed to be recovered [3, 4].

CNNs is a deep learning architecture for image and video data analysis. They use convolutional layers to scan the input data, identify patterns and features, and learn to classify objects in the images. They have achieved significant success in computer vision tasks, including object detection, image classification, and segmentation [5, 6]. In the same line, Recurrent Neural Networks (RNNs) are a type of deep learning architecture designed for processing sequences of data, such as time-series data, speech signals, and text [7]. The RNN is intended for sequence modeling and can be learned from historical input data. These characteristics make RNNs well-suited for natural language processing tasks, including language translation, text classification, and speech recognition [8, 9].

Deep learning models can autonomously learn and identify pertinent features from raw data, thereby minimizing the necessity for manual feature engineering. Additionally, deep learning models can be scaled to massive amounts of data, making them ideal for big data applications. Many researchers are beginning to use deep learning techniques to create models that recognize patterns and classify ransomware characteristics [10]. However, deep learning models also have some challenges. It requires large amounts of data and computational resources to train and can be challenging to interpret due to its complexity. It is also essential to be mindful of potential biases in the data and model, which can impact the accuracy and fairness of the results. Deep learning can transform numerous industries and applications [11] as the field progresses. This paper tries to build a deep-learning model to detect and classify ransomware attacks.

The paper is structured as follows: Section 2 summarizes recent studies on detecting ransomware attacks. Section 3 explains the implementation process of our proposed deep learning-based detection model. Section 4 discusses the results. Section 5 summarizes the paper, provides conclusions on the research findings.

II. Related Research

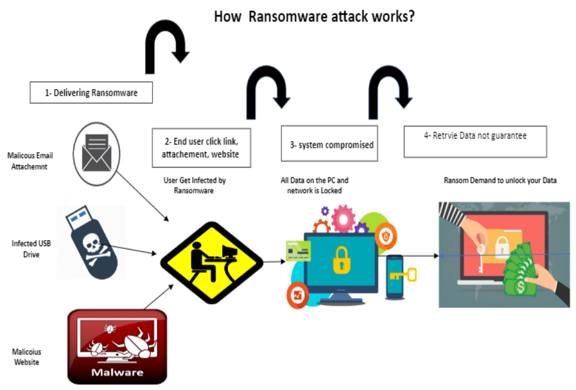
Several studies have revealed that ransomware attacks have impacted many IT critical infrastructures and individuals, which is why it has become the concern of many experts in the cybersecurity field to propose new approaches that have not been supported for further studies.

Ransomware is malicious software designed to launch harmful attacks on mobile devices. It is one of many malware categories that can compromise a victim's machine, including trojans, spyware, and more [12]. A ransomware attack is a cyberattack in which the attacker encrypts the victim's data and then demands a ransom, often in cryptocurrency, for the decryption key. The attack typically begins with the attacker gaining unauthorized access to the victim's system, for example, through a phishing email or vulnerable software [13]. Ransomware attacks can spread through various methods, including delivering its payload via malicious emails, circumventing standard access controls with Exploit Kits (EKs), injecting redirect links into JavaScript, drive-by downloads, waterhole attacks, and malvertising [14].

As a result, this form of malware has become a formidable threat, targeting individuals and organizations and leading to financial losses amounting to billions of dollars while affecting one million Android users in a single month. Once inside the system, the Ransomware infects and encrypts the victim's files, making them inaccessible. Following the attack, the perpetrator leaves a ransom note on the victim's computer, stipulating a payment demand in exchange for the decryption key. The ransom amount often increases if the victim does not pay within a specific timeframe. A workflow of the ransomware attack process is shown in Fig. 1.

Al-Hawawreh and Sitnikova [15] adapt CNN-based deep learning techniques for ransomware detection. The emulation sequence presents a specialized recurrent neural network for capturing local event patterns of Ransomware. Series using the concept of attention mechanism. It Demonstrates an improved (LSTM) model performance Sequence dataset obtained by ransomware emulation, an executable that targets the Windows environment. Malware detection under this investigation Ransomware is adaptable to specific structures. Ransomware characteristics can improve quality recognition by using ARI cells to direct attention to sequence inputs. However, the use of complex deep learning models may require high computational resources, making deployment in resource-constrained environments challenging.

Almomani, et al. [16] anticipated a novel approach for ransomware detection, which counters the shortcomings of traditional statistic-based malware detection methods, leveraging an evolutionary machine learning framework. This method integrates binary particle swarm optimization (BPSO) for feature selection and classification algorithm optimization, significantly enhancing the Support Vector Machine (SVM) algorithm's effectiveness by incorporating the Synthetic Minority Over-sampling Technique (SMOTE). Applied to a dataset of 10,153 Android applications, including 500 ransomware samples, this SMOTE-tBPSO-SVM technique has been shown to outperform conventional machine learning algorithms like K-NN, NB, MLP, and RF for sensitivity, specificity, and g-mean, achieving a remarkable 97.5% g-mean efficiency. However, the work poses potential overfitting due to the SMOTE technique when handling highly imbalanced datasets and the computational complexity introduced by the optimization of multiple hyperparameters, which may affect scalability.


How Ransomware attack works?
1- Delivering Ransomware
2- End user click link, attachement, website
3- system compromised
4- Retrvie Data not guarantee
Malicous Email Attachemnt
Infected USB Drive
Malicoius Website
Malware
User Get Infected by Ransomware
All Data on the PC and network is Locked
Ransom Demand to unlock your Data
Fig. 1 Ransomware attacks workflow.

A novel detection model that leverages a stacked Variational Autoencoder (VAE) combined with a fully connected neural network to uncover and learn the underlying system activity patterns. This model employs an extension method using VAE to generate additional training data, enhancing the model's ability to generalize and improve detection capabilities. The results demonstrated that this approach successfully detects ransomware activities, outperforming traditional machine learning algorithms with a 97.5% g-mean efficiency. Furthermore, the model shows promise in handling sparse data through data augmentation and managing unbalanced datasets, suggesting the potential for improved ransomware classification and identification in IIoT systems [17, 18].

Fernandez Maimo, et al. [19] automate the detection, classification, and mitigate ransomware with ICE in a real-time system. The proposed solution is the extension of the ICE++ architecture, which uses machine learning (ML) techniques for detection. The findings indicate that an effective strategy for mitigating ransomware involves isolating and replacing devices that have been infected. The approach demonstrated a 92.32% accuracy and 99.97% recall rate for detecting anomalies and an impressive 99.99% accuracy rate specifically for identifying ransomware. The designated methods, OC-SVM and Naive Bayes have proven adequate and detectable by achieving 99.99% classification accuracy with Naive Bayes. The feasibility of the proposed solution concerning time neutralizes ransomware attacks in 29.7 seconds with a spread of 63.1 seconds. However, the proposed classifier might hinder deployment in low-resource settings, and challenges in handling encrypted traffic, which limits the solution's effectiveness when ransomware encrypts its communication.

Khammas [20] discusses employing machine learning algorithms such as Decision Tree (J48), Random Forest, and Radial Basis Function (RBF) network, all integrated within the Waikato Environment for Knowledge Analysis (WEKA) framework, to identify potential ransomware threats. The results show that random forest provided the highest recognition accuracy, almost 98%. In the same line, Khammas [21] introduces a novel static analysis approach for detecting ransomware, leveraging machine learning techniques, specifically a random forest classifier, to analyze features extracted directly from raw bytes without complex decomposition processes. Experimental results demonstrate that the optimal configuration of a random forest classifier, with a tree count of 100 and a seed count of 1, achieves a high detection accuracy of 97.74%, an ROC of approximately 99.6%, and notably low false positive and negative rates.

III. Research Process Framework

The proposed model architecture is a CNN-based deep learning algorithm that trains the network to recognize data patterns indicative of ransomware attacks. CNN can be implemented by data preparation, defining the architecture, compiling, training, evaluating, optimizing, and deploying models. This model is designed to distinguish between ransomware and benign images. The code first imports necessary libraries and sets constants such as image dimensions prepares training data by categorizing ideas and defines the layers of the CNN model. It then prepares test data and trains the model using early stopping and model checkpointing. Finally, it specifies the batch size and runs the training process for a set number of epochs.

A. Formulation of The CNN Algorithm

The equations for convolution operation can be represented as shown in (1):

$$f[I, J] = (g * h)[I, J] = \sum g[m, n] * h[I - m, j - n] \quad (1)$$

where f[i , j] is the value at position (i , j) in the feature map, g is the input image, h is the filter, and m and n are the indices used for summation.

The equation for max pooling can be represented as illustrated in (2):

$$f[i, j] = max(g[i * stride : I * srtide + pool\_size, j * stride : j * stride + pool\_size]) \quad (2)$$

Where f[i , j] is the output value at position (i, j) in the pooled feature map, g is the input feature map, the stride is the stride length (i.e., the number of pixels to skip between each group), and pool_size is the size of the pooling window.

The equation for ReLU can be represented as shown below in (3):

$$f(x) = \max(0, x) \quad (3)$$

The equation for the fully connected layer can be represented as in the (4):

$$f(x) = \text{softmax}(Wx + b) \quad (4)$$

In this context, W represents the matrix of weights, x denotes the flattened output from the preceding layer, b stands for the bias vector, and the softmax function is applied to normalize the scores, resulting in a probability distribution.

B. Implementation of The Proposed Model Design

The designed model architecture features a sequence of CNN blocks, incorporating five layers plus an additional fully connected layer at the end. the initial block introduces a conv2d layer with 32 filters, each measuring 3x3, utilizing a relu activation function, and applying "same" padding to maintain the input size. the model accepts input as image_width, image_height, image_channels with image_width and image_height denoting the dimensions of the input image and image_channels indicating the color channels count (3 for RGB images). subsequently, a maxpooling2d layer with a 2x2 pool size and a 2x2 stride follows, effectively halving the spatial dimensions of its input, thereby streamlining further processing. Fig. 2 shows a layered diagram of the CNN model.

The subsequent blocks follow a similar pattern, with each successive block doubling the number of filters. The second block has 64 filters, the third block has 128, and the fourth and fifth blocks have 256 filters each. This transformation allows the output of the convolutional layers to be fed into a fully connected layer. Following this, the flattened output is directed through a dense (fully connected) layer that contains 512 units, utilizing a ReLU activation function for processing. This layer acts as a classifier, using the learned features from the convolutional layers to make predictions about the input image. To mitigate overfitting, the model includes a Dropout layer with a dropout rate 0.5. This configuration is designed to produce the probabilities indicating the likelihood of the input image being classified into one of the two possible categories. This model architecture is a solid choice for image classification tasks, as it allows for learning and extracting valuable features from the input images while minimizing overfitting. Overall, the proposed model architecture balances depth and complexity well, making it practical for various image classification tasks.

C. Algorithm Description

CNN is a type of deep learning algorithm that is well-known for image classification. Drawing inspiration from the mammalian visual cortex, CNNs are engineered to autonomously identify and learn layered visual feature representations directly from unprocessed image data. In addition, Once trained, the CNN can classify new images by feeding them through the network and computing the output probabilities using the SoftMax layer. The class with the highest chance is then taken as the final prediction. Furthermore, a CNN classifies images of malware as benign based on parameters. It starts by importing the necessary libraries and setting some constants, such as the image size and the number of channels.

It then prepares the training data by reading the images, assigning categories based on the file names, and creating a

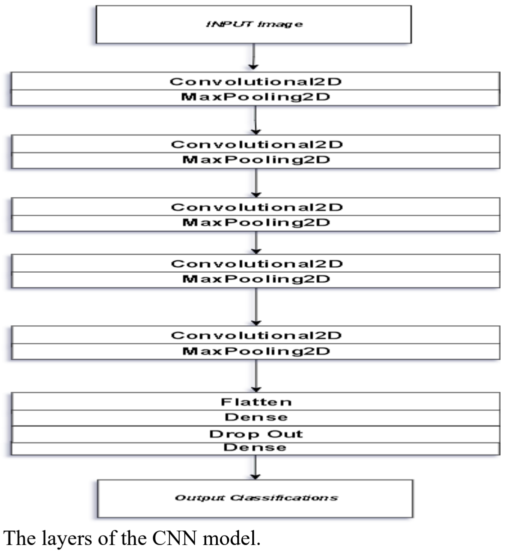


Fig. 2 The layers of the CNN model.

data frame to hold the information. The flowchart of the CNN algorithm is shown in Fig. 3.

The CNN model uses Keras, with various convolutional and pooling layers, a dense layer, and a softmax output layer. It is compiled with the Adam optimizer and trained using data augmentation techniques. The code also includes callbacks for early stopping, learning rate reduction, and an image generator and evaluates the model using various performance metrics.

D. Dataset Description

The dataset used for ransomware detection consists of malicious and benign executable files that were converted into grayscale images for classification using CNN. The main features of the dataset include. The exe files of benign were from 1986, while the ransomware files were from 1829. When we convert exe files to grey-scale images, it will have more than one image, which results in a total of 193033 that will be used on the CNN model. Total dataset (post-augmentation): Augmentation could easily multiply the dataset by 4x to 10x, resulting in 4000 to 20,000 image samples for training and testing. The executables were converted into grayscale images using the PIL, enabling image classification methods to be applied for malware detection. The dataset was processed on both Windows 11 and Kali Linux due to system restrictions on executing the malware for data conversion. Fig. 4 illustrates an example of benign ransomware impacted by Image data augmentation, a method employed to expand the dataset size by generating altered versions of the initial images. This entails utilizing diverse transformations or manipulations on the images, including rotation, translation, scaling, and flipping. By augmenting the dataset with these modified images, we can provide more variations to the model during training, making it more robust and less prone to overfitting, improving the model's accuracy.

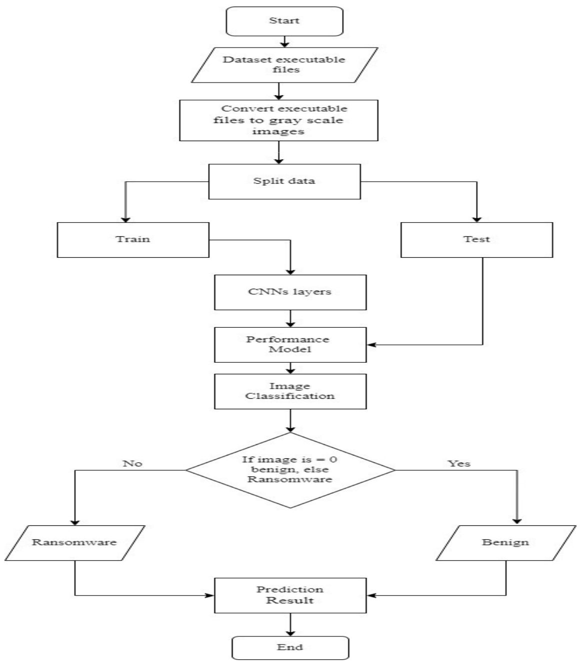


Fig. 3 Flowchart of the CNN algorithm.

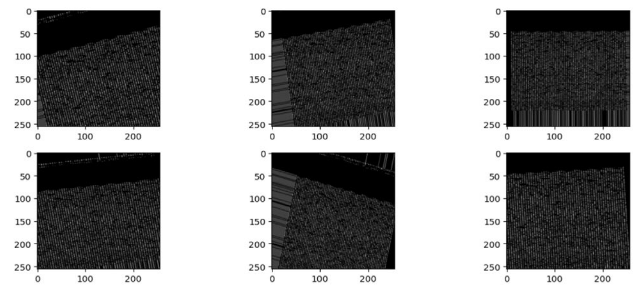


Fig. 4 Image augmentation techniques.

E. Experimental Setup

The experimental setup simulated two operating systems, which are Linux and Windows; since the dataset contains malware and benign executables, unfortunately, the Windows 11 protections stop the execution process of converting binaries to images, then migrating to Kali Linux to complete the converting executables to images, the tool can run by Graphics Processing Unit (GPU), Central Processing Unit (CPU), and Tensor Processing Unit (TPU), then moving back to Windows 11 to complete the process using Anaconda which is a platform that provides much software for developers such as VS Code, Juputer, and PyCharm, etc.

F. Performance Metrics

The proposed CNN-based deep learning model is evaluated on F1 score, Precision, Recall, and Accuracy performance metrics. The results are analyzed and shown in the figures for comparison purposes.

Precision*:* Precision assesses the model's accuracy in positive prediction. The equation of precision is demonstrated in (5):

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

Recall: Recall is the fraction of actual positive cases that the model correctly identifies, calculated by dividing the count of true positives (TP) by the total of true positives and false negatives (FN). The equation of Recall is presented in (6):

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

F1 Score: The F1 Score is calculated as the harmonic mean between precision and recall, achieved by doubling the multiplication of precision and recall, and then dividing that by the addition of precision and recall. The equation of F1 is displayed in (7):

$$F1 = 2 \times \left(\frac{Precision \times recall}{Precision + recall}\right) \tag{7}$$

Accuracy: Accuracy signifies the proportion of correct predictions generated by the model among all predictions made. Accuracy is calculated by dividing the total of true positives and true negatives (TN) by the overall count of true positives, false positives, true negatives, and false negatives, offering a comprehensive assessment of the model's predictive accuracy. The equation of accuracy is displayed in (8):

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \quad (8)$$

IV. Results and Discussion

The whole dataset consists of ransomware and benign images that have been used to apply a CNN algorithm to distinguish and detect a ransomware attack by analyzing the patterns of the dataset after splitting the data, training, and evaluating the model. The experimental tool was conducted through Python libraries of machine learning. As illustrated in Fig. 5 below, the CNNs performance is provided in accuracy, precision, F1 score, and Recall. We have listed both results to compare our model's training and validation results. The training accuracy is 0.9476, slightly more significant than the validation of accuracy, which is 0.9383 percent. This implies that accuracy measures the percentage of correctly classified examples out of the total number of models in the test set in CNNs. This metric is commonly used to evaluate the overall performance of a CNN.

Fig. 6 shows the model's accuracy learning process consisting of training and validation accuracy; the x-axis represents the number of epochs that refers to the model's accuracy on the training data during the training process, while the y-axis represents the percentage of accuracy on each phase or age that refers to the model's accuracy, it can be observed that training accuracy has started from 0 – 5 epoch with the same rate, which is 0.8763, whereas, the validation accuracy began after one epoch of training, it was also stable from 0 – 5 epoch with 0.8830. what is more, the training accuracy has gradually increased until it reaches 0.9476, while the validation accuracy Ascends and descends till it reaches 0.9383. however, there was a sudden decrease in the validation accuracy at the twenty-nine epochs, which immediately increased on the last epoch of the learning process.

Fig. 7 shows the model's loss of the learning process, which consists of training and validation loss. The x-axis indicates the epoch count, corresponding to the model's training accuracy throughout the training period. Meanwhile, the y-axis shows the accuracy percentage for each phase or epoch, reflecting the model's performance accuracy.

The training loss starts at 0.3834 percent, while the validation loss starts at 0.3378. The training loss has the same result until epoch five, while the validation loss decreases at epoch nine from 0.3371 to 0.2358. after epoch five, the training loss was dramatically reduced until it reached 0.1574 compared to the validation loss. It Climbed up quickly from 0.3371 to 0.8509 and then declined suddenly to 0.2569. This connotes that the error between the model's predicted output and the actual production is calculated using the training data. During training, the model iteratively adjusts its weights and biases to minimize this error and improve its accuracy in predicting the output. With ongoing training, an ideal scenario would see a reduction in training loss, signifying an improvement in the model's ability to represent the training data accurately.

Fig. 8 shows the AUC measurement of the learning process, which consists of training and validating the AUC rate. The x-axis signifies the number of epochs that refer to the model's accuracy on the training data during the training process. In contrast, the y-axis denotes the accuracy percentage on each phase or epoch. The AUC training rate was less than the validation rate from the first epoch. The Training rate increased from the first epoch until the end of the learning process, reaching 0.9837 percent. In comparison, the Validation AUC slightly increased and decreased until age twenty-eight, suddenly declined until it reached 0.7397, and then sharply increased to 0.9754 percent.

AUC can be calculated for both the training and validation datasets. The training AUC measures the model's ability to classify examples from the training set accurately.

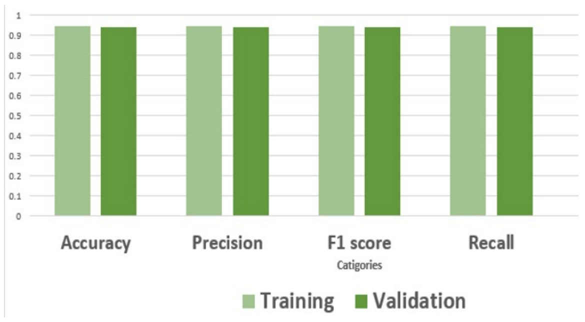


Fig. 5. Performance of the CNNs parameter.

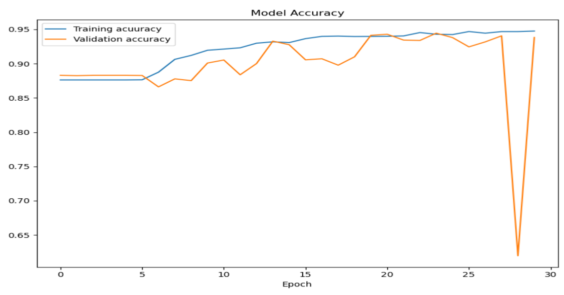


Fig. 6 Accuracy for training and validation.

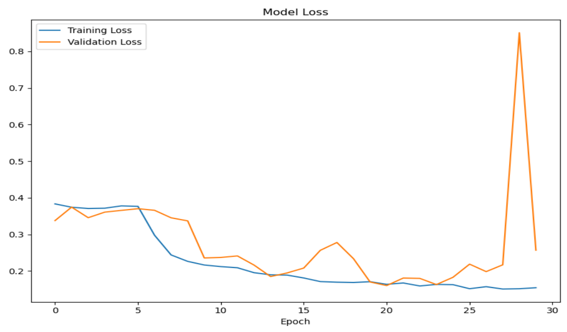


Fig. 7 Model loss.

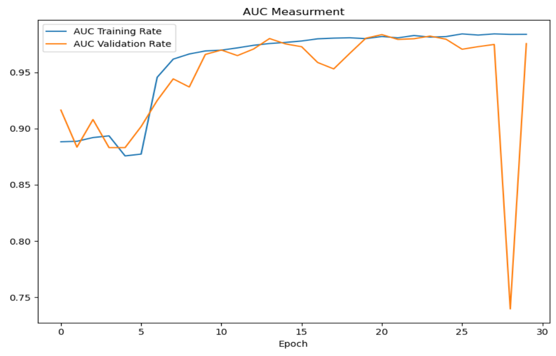
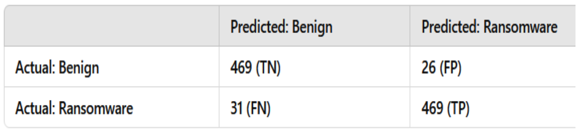


Fig. 8 The AUC Measurement for training and validation.

In contrast, the validation AUC measures its ability to simplify new, unseen data in the validation set.

Fig. 9 shows the confusion matrix which is based on the values from the precision, recall, and assumed class distribution.

Fig. 10 shows the standard parameters that will be compared between the CNNs model and an automated malware pattern extraction (early detection tool) tool for specific ransomware attacks that have been detected.



	Predicted: Benign	Predicted: Ransomware
Actual: Benign	469 (TN)	26 (FP)
Actual: Ransomware	31 (FN)	469 (TP)

Fig. 9 The confusion matrix.

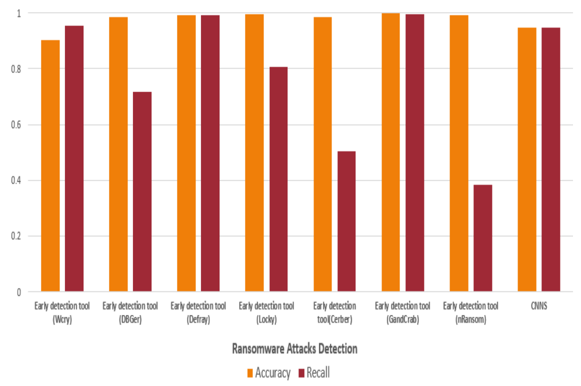


Fig. 10 The standard parameters evaluation.

V. Conclusion

This research advances our understanding of the critical role machine learning and deep learning play in bolstering cybersecurity, mainly through ransomware detection. By meticulously selecting the CNN for its renowned image classification capabilities and self-learning efficiency, the study bridges a crucial gap in existing cybersecurity methodologies, showcasing a novel approach to identifying and mitigating cyber threats. Through a detailed examination and practical application involving developing a Python-based framework to train and validate the CNN model on grayscale images of executable files, the research demonstrates the model's high accuracy and recall rate of 0.9476. This validates the effectiveness of the CNN model against sophisticated cyber threats, including elusive zero-day attacks, and highlights its potential to enhance current cybersecurity defenses significantly.

References

[1] A. Alraizza and A. Algarni, "Ransomware detection using machine learning: A survey," Big Data and Cognitive Computing, vol. 7, no. 3, p. 143, 2023.

[2] S. Kok, A. Abdullah, N. Jhanjhi, and M. Supramaniam, "Ransomware, threat and detection techniques: A review," Int. J. Comput. Sci. Netw. Secur, vol. 19, no. 2, p. 136, 2019.

[3] M. Musonda, A. Zimba, and M. Sinyinda, "Machine learning-based crypto ransomware detection model on windows platforms," in Proceedings of International Conference for ICT (ICICT)-Zambia, 2023, vol. 5, no. 1, pp. 141-147.

[4] J. A. Gómez Hernández, P. García Teodoro, R. Magán Carrión, and R. Rodríguez Gómez, "Crypto-Ransomware: A Revision of the State of the Art, Advances and Challenges," Electronics, vol. 12, no. 21, p. 4494, 2023.

[5] G. O. Ganfure, C.-F. Wu, Y.-H. Chang, and W.-K. Shih, "Deepware: Imaging performance counters with deep learning to detect ransomware," IEEE Transactions on Computers, vol. 72, no. 3, pp. 600-613, 2022.

[6] L. E. Haddad, M. Hanoune, and A. Ettaoufik, "Computer Vision with Deep Learning for Human Activity Recognition: Features Representation," Engineering Applications of Artificial Intelligence, pp. 41-66, 2024.

[7] L. S. Vidyaratne, M. Alam, A. M. Glandon, A. Shabalina, C. Tennant, and K. M. Iftekharuddin, "Deep cellular recurrent network for efficient analysis of time-series data with spatial information," IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 11, pp. 6215-6225, 2021.

[8] S. Das, A. Tariq, T. Santos, S. S. Kantareddy, and I. Banerjee, "Recurrent neural networks (RNNs): architectures, training tricks, and introduction to influential research," Machine Learning for Brain Disorders, pp. 117-138, 2023.

[9] S. i. M. Abdulhamid, N. A. Sadiq, M. Abdullahi, N. Rana, H. Chiroma, and D. E. Gbenga, "Development of blowfish encryption scheme for secure data storage in public and commercial cloud computing environment," 2018.

[10] D. W. Fernando, N. Komninos, and T. Chen, "A study on the evolution of ransomware detection using machine learning and deep learning techniques," IoT, vol. 1, no. 2, pp. 551-604, 2020.

[11] R. A. Khalil, N. Saeed, M. Masood, Y. M. Fard, M.-S. Alouini, and T. Y. Al-Naffouri, "Deep learning in the industrial internet of things: Potentials, challenges, and emerging applications," IEEE Internet of Things Journal, vol. 8, no. 14, pp. 11016-11040, 2021.

[12] M. Gopinath and S. C. Sethuraman, "A comprehensive survey on deep learning based malware detection techniques," Computer Science Review, vol. 47, p. 100529, 2023.

[13] A. Zimba and M. Chishimba, "On the economic impact of crypto-ransomware attacks: The state of the art on enterprise systems," European Journal for Security Research, vol. 4, no. 1, pp. 3-31, 2019.

[14] U. Zahoora, M. Rajarajan, Z. Pan, and A. Khan, "Zero-day ransomware attack detection using deep contractive autoencoder and voting based ensemble classifier," Applied Intelligence, vol. 52, no. 12, pp. 13941-13960, 2022.

[15] M. Al-Hawawreh and E. Sitnikova, "Leveraging deep learning models for ransomware detection in the industrial internet of things environment," in 2019 military communications and information systems conference (MilCIS), 2019: IEEE, pp. 1-6.

[16] I. Almomani et al., "Android ransomware detection based on a hybrid evolutionary approach in the context of highly imbalanced data," IEEE Access, vol. 9, pp. 57674-57691, 2021.

[17] M. Al-Hawawreh, M. Alazab, M. A. Ferrag, and M. S. Hossain, "Securing the Industrial Internet of Things against ransomware attacks: A comprehensive analysis of the emerging threat landscape and detection mechanisms," Journal of Network and Computer Applications, p. 103809, 2023.

[18] A. T. Salim and B. M. Khammas, "Simplified review on cyber security threats detection in IoT environment using deep learning approach," Journal of the College of Basic Education, vol. 29, no. 119, pp. 49-22, 2023.

[19] L. Fernandez Maimo, A. Huertas Celdran, A. L. Perales Gomez, F. J. Garcia Clemente, J. Weimer, and I. Lee, "Intelligent and dynamic ransomware spread detection and mitigation in integrated clinical environments," Sensors, vol. 19, no. 5, p. 1114, 2019.

[20] B. M. Khammas, "Comparative analysis of various machine learning algorithms for ransomware detection," Telkomnika (Telecommunication Computing Electronics and Control), vol. 20, no. 1, pp. 43-51, 2022.

[21] B. M. Khammas, "Ransomware detection using random forest technique," ICT Express, vol. 6, no. 4, pp. 325-331, 2020.