Unmasking Malware with MDCNN: A New Era of Image-Based Detection

Conference Paper · January 2025

DOI: 10.1109/ICMSCI62561.2025.10894516

Gomathy M.
BCIIT,IP University
5 PUBLICATIONS **6** CITATIONS

Unmasking Malware with MDCNN- A New Era of Image Based Detection

Gomathy M
Research Scholar, School of Computing Sciences, Vels Institute of Science, Technology and Advanced Studies (VISTAS)
Chennai, Tamilnadu
marimuthu.gomathy@gmail.com

A. Vidhya
Assistant Professor, Department of Information Technology, School of Computing Sciences, Vels Institute of Science, Technology and Advanced Studies (VISTAS)
Chennai, Tamilnadu.
avidhya.scs@velsuniv.ac.in

***Abstract*— Cybercrime has been a worldwide issue since the internet's inception. The various online crimes that take place have an impact on the general public. The various forms of cybercrime committed online and around the world are always evolving due to the development of the internet. In the current digital age, when malware diversity and volume are increasing rapidly, new methods must be used to identify malware more quickly and accurately. Malware, which includes worms, trojans, spyware, and adware, can have serious repercussions, including financial losses, data breaches, and the interruption of essential services. Manual heuristic inspection in malware analysis is neither efficient nor effective in keeping up with the rapid spread of malware or in analyzing new infections. Deep learning enhances automatic malware variant detection and classification as it provides better categorization by building neural networks with more potentially different layers. They have been applied to automate investigations using static and dynamic analysis, where malware with comparable behaviors is grouped and unidentified malware is categorized into families according to how close it is to the other malware. In this research paper, we extract various feature sets from malware image files, including patterns, sizes, and textures, we utilized a novel approach, the Malware Detection Convolutional Neural Network (MDCNN) was employed to analyze malware in image files and the proposed model shows 96.83% accuracy rate in classifying the malware according to its family. The efficiency of the model is compared with other deep learning models and has shown the proposed model outperforms the other models**.

Keywords— Cybercrime, CNN, Deep Learning Images, Malware,

I. INTRODUCTION

Malware can spread through multiple formats, such as text files, PE headers, and images. Detecting malware in any of these forms is essential for protecting against cyber-attacks. This paper focuses on analyzing malware that is represented in image format. We present a graphical representation showing that the Trojan malware family accounts for 71% of newly emerged malware varieties from the internet [1]. Malware consists of malicious code and various system calls that indicate unauthorized access in different areas. Identifying and classifying malware image patterns are a challenging task for antivirus vendors. Within the field of cybersecurity, malware research typically focuses on executable files, using a variety of methods to identify and eliminate risks. Nevertheless, as cyber dangers change, attackers are turning to less conventional means of getting past defenses. Steganography, a specific technique used to embed spyware within image files, is a notable method. Through investigating the embedding of malware into images, researchers aim to enhance detection techniques, bolster digital forensics capabilities, and prevent data exfiltration. Gaining knowledge of these methods can assist in safeguarding sensitive data and improve defenses against Advanced Persistent Threats (APTs) [3]. Thus, identifying spyware in images is essential to preventing cyber-attacks and guaranteeing thorough security protocols. In this paper, we introduce an innovative method called Malware Detection Convolutional Neural Network (MDCNN) for analyzing malware. This approach involves converting executable files into image files, which are then analyzed using image processing and deep learning techniques. This transformation enables us to identify and classify malware, offering a new perspective compared to traditional analysis methods. By employing this technique, our goal is to uncover concealed patterns and signatures that might not be detected in traditional executable analysis, ultimately enhancing the efficiency of malware detection and response strategies. Section II discusses various authors' points on image analysis in malware. Section III provides an in-depth description of images related to malware and its various sections. An extensive knowledge of CNNs' use in malware investigation is given in Section IV, which also describes their design for analyzing image data. Section V introduces the innovative method MDCNN for analyzing malware within images, utilizing an image dataset sourced from VirusTotal. Section V provides the experimental outcomes of using the MDCNN model to analyze malware in image form.

II. II LITERATURE REVIEW

The author of this study proposes a unique Reversible Data Hiding (RDH) strategy called "GRDH" that is based on the GAN model in the article [1]. A strong image generator that could create realistic images was trained using the GAN model. The trained generator was then fed into the CycleGAN model to produce images with distinct semantic information. The author created a mapping relationship between noise and messages to embed messages. By teaching an extractor to eliminate noise from the final dense image, the message can be extracted. The suggested method's efficiency is demonstrated by the outcomes of the experiments. It is possible to disguise binary data in images by utilizing StegnoGAN, which was proposed by [2]. Several unique models have been proposed for image production. These models are the basic, residual, and dense encoder architectures. It is recommended that the Reed-Solomon bits-per-pixel (RS- BPP) be utilized to better evaluate the embedded data capacity while using GAN models for steganography. An example of an ISGAN model was presented by [3] to disguise the greyscale image within the Y channel values of the color image. All of these components are included to improve the overall performance of the security system. In the ISGAN model, CNN-based layers are utilized by three of the networks that comprise the model. In a research article [4] proposed the use of ACGANs as a means of preventing the cover images from being altered while secret data was being transmitted. There is a set of labels and the image database that are taken into consideration when selecting the image that will be used for the cover. The sender uses a dictionary to map the intended message to the labels, and then they use the labels to select an image from the database to send data.

The research paper [5] proposed the HiGAN model, which is based on GANs, as a technique for hiding images inside other images. A color image is concealed inside another color image using the HiGAN technique. Three sub networks are used by HiGAN to carry out GAN-based steganography. The color secret image must be hidden, and the encoder does this by using a correspondingly sized cover color image.

Convolutional neural networks (CNNs) form the foundation of the novel ensemble architecture that the author [6] presented for effective malware detection, both packed and unpacked. By using a variety of CNN architectures, it was possible to extract features with better quality than with traditional methods. They increased the classification accuracy by obtaining generic features through the use of transfer learning. However, this approach has a disadvantage as it requires more time to execute.

To help CNNs deal with skewed data and difficulties in malware picture classification, [7] devised a weighted softmax loss function. Through the implementation and fine-tuning of this weighted softmax loss in the deep CNN model, the author achieved encouraging outcomes in the malware picture classification.

The study in the article [8] suggests a method for classifying malware that makes use of convolutional neural networks (CNNs) and image processing. The process entails collecting Local Binary Pattern (LBP) features from malware and translating it into binary pictures. The infected images are first split into 3x3 grids, and then LBP is used to extract features that can be used to classify textures or patterns.

The author of [9] used Siamese neural networks to classify malware images using one-shot image recognition. The outcomes of the experiment showed that this strategy worked better than the baseline techniques. According to empirical findings, Siamese network-based one-shot image recognition produced a test accuracy of 92.0%.

A classification technique based on common visual attributes is presented in this research. The author [10] used K-nearest neighbor (KNN) for classification and GIST filtering, which efficiently computes texture features for malware images using wavelet decomposition. With a total classification time of 1.4 seconds, the GIST feature extraction took 54 ms. Comparing the suggested method to previous classification and filtering algorithms, it is about 40 times faster.

[11] proposed a 98% accurate Convolutional Neural Network (CNN) model for classifying malware images. According to the experimental findings, this accuracy is comparable to the k-nearest neighbor (KNN) classification method. This strategy works better than several conventional classification techniques.

To categorize malware into its appropriate family and differentiate it from benign files, the author presents Malware Spectrogram Image Classification (MSIC), which makes use of spectrogram images in conjunction with convolutional neural networks (CNNs). The malware file is shown as a spectrogram image in the first stage, and this image is used as input for the CNN classifier. CNN is used in the second stage to categorize the spectrogram images, and it works better than any other technique.

A deep learning-based approach for malware detection (DLMD) that classifies various malware types utilizing static approaches was proposed [12]. The suggested method uses two different Deep Convolutional Neural Networks (CNNs) to extract features from byte data. The drawbacks of individual feature areas are lessened with the aid of this hybrid feature space. Eventually, a Multi-Layer Perceptron (MLP) is used to classify malware families.

The performance of Deep Neural Networks (with various architectures) and traditional machine learning Random Forest techniques for the classification of malware with various feature sets is compared in this article by [13]. Overall, it was shown that Random Forest performed better than DNN's multilayer architecture. RF and DNN achieved the highest accuracy of 99.78% and 99.21%, respectively.

To identify malware families in a metamorphic malware environment, this paper [14] presents a deep convolutional neural network (CNN) model supplemented by image augmentation. The author's main contribution is the creation of a malware family classification model that uses CNNs to improve image classification accuracy and data augmentation to handle malware variants. Malware developers frequently utilize a variety of strategies to mask their destructive activity, and data augmentation is a useful tool for overcoming these difficulties.

[15] Investigates picture classification with TensorFlow-implemented deep neural networks (DNNs), or deep learning. TensorFlow is a well-known deep learning toolkit [16] used to classify the MNIST dataset. The impacts of different activation functions on classification performance were also compared in the study. On the test data, the ReLU activation function showed a classification accuracy of 98.43%.

III. Malware Detection Convolutional Neural Network(MDCNN)

A) Malware Images

Identifying visual anomalies in images generated from executable files can be more straightforward than examining their textual or binary content. When these files are transformed into grayscale images, each byte maps to a pixel's intensity, unveiling patterns and structures that might be obscured in their raw forms [17]. This visual approach can expose irregularities like unexpected patterns or sudden shifts in pixel intensity, which could signal the presence of malicious code. Malware often employs obfuscation techniques that leave unique visual signatures, which deep learning can be trained to detect [18][23]. By leveraging these grayscale images, cybersecurity systems can improve Trojan detection's precision and effectiveness.

To analyze grayscale images effectively and classify them into different malware classes, an innovative technique named Malware-Detection Convolutional Neural Network (MDCNN) has been utilized. MDCNN is an abbreviation for Malware Detection Convolutional Neural Network and it is a new multi-class CNN architecture for the specific application of malware detection. It employs the virtues of CNNs [19][24] to automatically detect and retrieve hierarchically the features from images, and thus, it decreases the number of potential hypotheses and increases the accuracy of the prediction. The network architecture of MDCNN is designed to process the multi-dimensional data represented by grayscale images, capturing both spatial and intensity-based features that are crucial for distinguishing between different types of malwares [25]. The primary objective of using MDCNN is to perform multi-class classification of malware based on the visual features extracted from grayscale images. This approach enables cybersecurity analysts to categorize and prioritize threats based on their specific characteristics and behaviors, facilitating more targeted and effective mitigation strategies. By employing advanced deep learning techniques such as MDCNN, cybersecurity systems can enhance their ability to accurately classify and mitigate various malware, contributing to stronger defenses against cyber threats.

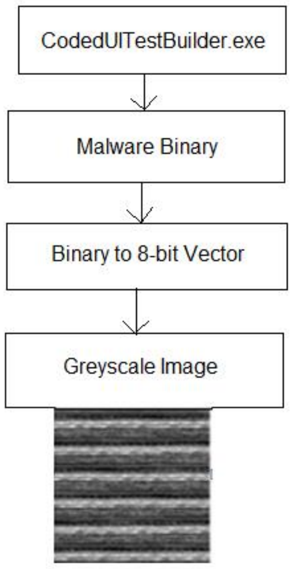

Figure 1 Depicting Malware as an Image

To convert a binary file into an 8-bit vector and then into grayscale images, begin by reading the file's binary content using Python's file handling capabilities in binary mode ('rb'). This involves opening the file and reading its contents into a variable. Next, use NumPy to convert the binary data into an array of 8-bit integers (uint8), where each byte in the binary file is equivalent to a pixel's intensity value ranging from 0 to 255. This array serves as the foundational 8-bit vector representation of the grayscale image. Reshape the 8-bit vector into a 2-dimensional array that reflects the desired dimensions of the grayscale image. This step involves determining the width and height of the image based on the binary file's size and structure. With the reshaped array ready, you can use libraries such as PIL (Python Imaging Library) to create a grayscale image (mode='L'). Finally, save the generated grayscale image in a suitable format like PNG or JPEG. Figure 1 shows the visual conversion of an executable file to a grey-scale image. This conversion process enables visual representation and analysis of binary data, facilitating tasks such as malware detection through image-based machine learning approaches or simple visual inspection by cybersecurity analysts [20]-[22]. Figure 2 displays a number of grayscale images that depict malware executable files that have been transformed. The visual comparison between various malware kinds is made possible by the inclusion of samples from the benign class and the five other malware families in these images.

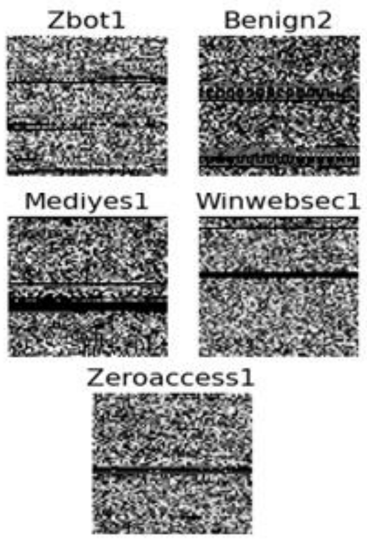

Figure 2 Malware Greyscale Images

The data for this analysis was sourced from Virus Total. Out of the 7630 executable files obtained, all were converted into grayscale images for malware analysis. After pre-processing, 6985 high-quality grayscale images were selected for further analysis. Figure 3 shows the pictorial representation total number of malware present in each family.

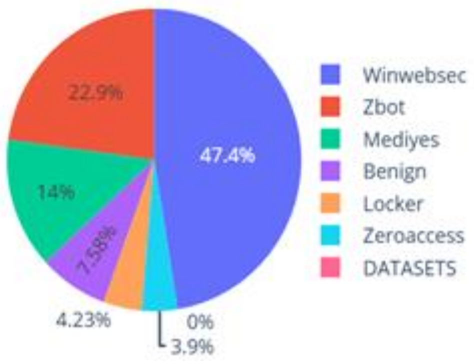

Figure 3 Percentage Distribution of Images Across Malware Families and Benign Files

The dataset used for image-based malware analysis is organized into several folders, each containing a distinct number of files that represent different types of malware and benign samples. These folders include Locker1 with 231 files, Mediyes1 with 1015 files, Winwebsec1 with 3090 files, Zbot1 with 1498 files, Zeroaccess1 with 486 files, and Benign2 with 665 files as shown in the above figure.

B. Unveiling the Architecture of MDCNN: A Deep Dive

The method of analyzing and categorizing malware utilizing greyscale images generated from ransomware executables and a Malware Detection Convolutional Neural Network (MDCNN) model is depicted in the architecture diagram above. Malware Detection Convolutional Neural Network (MDCNN) used for image categorization. It is built using the Keras framework and employs several of layers to alter the input images into output class probabilities. Convolutional layers, pooling layers, flattening layers, fully connected layers, and dropout layers are among the layers in the design, and each has a distinct function in the training process.

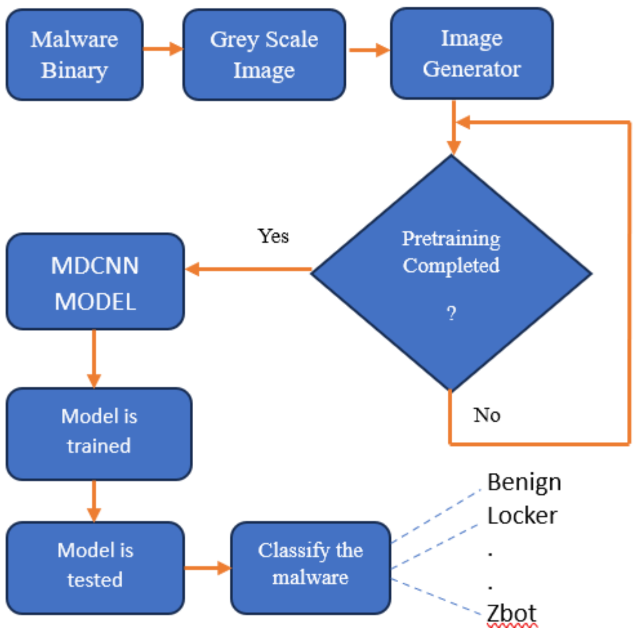

Figure 4 Architecture of MDCNN Model

Figure 4 shows the architectural diagram of the proposed model MDCNN. First, executable files are transformed into malware binaries, which are subsequently transformed into greyscale pictures. The analysis of visual patterns in the malware is made possible by the fact that each pixel in these pictures corresponds to a byte in the binary file. These greyscale pictures are preprocessed using an image generator. Techniques including scaling, normalization, and data augmentation may be used in the preprocessing. The completion of the pretraining procedure is verified by the decision point. If not, the procedure repeats, creating new images and undergoing preprocessing until the pretraining is finished. An MDCNN model receives the greyscale images that have been preprocessed. This model probably consists of several convolutional layers intended to extract patterns and spatial hierarchies from the images. The model is composed of at least three hidden layers, each of which introduces non-linearity through the use of ReLU (Rectified Linear Unit) activation functions, allowing the MDCNN to learn complicated patterns. A Sparse Categorical Cross-entropy loss function is applied to the training process to quantify the difference between the predicted and real labels. For classification jobs where the output labels are integers, this loss function is appropriate.

Following training, the model is assessed by grouping images into families based on the type of malware they include (e.g., Benign, Locker, Zbot). To verify that the model has learned to distinguish between various malware families, its performance is assessed to determine its ability to classify new images. Overall, the architecture outlines a process where malware binaries are converted into grayscale images, preprocessed, and then fed into an MDCNN for training and classification. The model MDCNN is capable of classifying the malware based on the visual patterns identified in the grayscale images, using deep learning techniques with an accuracy score of 96.83%.

c) Pseudocode MDCNN model

```
•   Input: Malware Images
•   Output: Predicting the images and the family of Malware
Step 1: Initialize and Import the necessary Libraries
Step 2: Load the dataset and convert to greyscale images.
Step 3: Apply Pre-processing using the image generator on all the images
Step 4: Split the dataset into training and validation
Step 5: Input the dataset to the chosen MDCNN model for training
Step 6 : Call convolution_multiple_filters(I, filters, S, P)
Step 7: Apply Adam Activation and Sparse cross Entropy loss function on the features extracted
Step 6: Model classifies and Predicts the output.
Step 7: Test the model
Step 8: Model Classify the image to its appropriate Malware Family
```

<u>Algorithm for Convolutional layer to combine the features extracted</u>

```
    • Input:  Image I, Filters F, Stride S, Padding P
    • Output: Extract the features and combine it
Algorithm convolution_multiple_filters (I, filters, S, P)
Step 1: Initialize I, filters, S, P
Step 2:   if P > 0:
padded_image = np. zeros((I.shape[0] + 2 * P, I.shape[1] + 2 * P))
padded_image [P:-P, P:-P] = I
I=padded_image
 Step 3: //Output dimensions
H_out = (I. shape [0] - filters [0]. shape [0] + 2 * P) / (S + 1)
W_out = (I. shape [1] - filters [0]. shape [1] + 2 * P) / S + 1
Step 4: Set feature_maps = np.zeros((H_out, W_out))
Step 5:   for k, F in enumerate(filters):
            for i in range(H_out):
            for j in range(W_out):
            R = I [i*S : i*S + F. shape[0], j*S: j*S + F.shape[1]]
            feature_maps[k][i, j] =  np.sum(R * F)
Step 6: set feature_maps = np.maximum(0, O)
Step 7:   combined_output = np.stack(feature_maps, axis=-1)
            return feature_maps, combined_output
```

IV Result and Analysis of MDCNN

The standard metrics used to assess the effectiveness of models for classification are accuracy, precision, and F1 score.

<u>Accuracy</u>:

The percentage of accurately identified events—both true positives and true negatives is known as accuracy. The accuracy score evaluates how well the model predicts the dataset's categories. A higher accuracy rate suggests that the model is more frequently producing accurate predictions. The formula for calculating the accuracy is given by equation 1.

$$Accuracy = \frac{(TruePositive + TrueNegative)}{TruePositive + TrueNegative + FalsePositive + FalseNegtive} \quad (1)$$

Precision

Precision gauges how well a model prevents false positives or the incorrect classification of harmless files as malicious. Improved precision results in fewer false positives, which is essential for applications due to misidentifying innocuous files as malware might cause needless warnings or actions. The formula for calculating the precision is given by equation 2.

$$precision = \frac{TruePositive}{TruePositive+FalsePositive} \quad \textbf{(2)}$$

F1 Score

The F1 score provides an appropriate ratio between precision and recall by taking the harmonic mean of these two parameters. The F1 score is at its lowest at 0 and its highest at 1 (perfect recall and precision). It provides a single statistic for assessing model performance by striking a compromise between recall and precision. F1 score and recall are calculated by Equation 3 and Equation 4.

$$F1\ Score = \frac{2X(PrecisionXRecall)}{Precision+Recall} \quad (3)$$

$$Recall = \frac{TruePositive}{TruePositive+FalseNegative} \quad (4)$$

The following table presents the performance metrics of the MDCNN model on the test data, derived from the computations outlined above.

Table 1 Comparison of Performance Score of MDCNN

Comparison of Performance Score of MDCNN for Malware Analysis with Deep Learning Model in Percentage				
Model	Accuracy	F1 Score	Recall	Precision
VGG16	94.2	93.23	91	94.5
INCEPTION	93.1	92.14	89	88.6
MDCNN	96.83	97.56	96.78	97.3
RESNET	94.56	94.4	92	93.2

The MDCNN model is compared with the standard CNN model to show its performance level. Table 1 illustrates the comparative performance levels of various CNN models. The accompanying table details the accuracy, F1 score, and recall values achieved by each model. The VGG16 model, 93.1% for the Inception model, 94.56% for the ResNet model, and 96.83% for the novel MDCNN model have the highest accuracy scores. Comparably, VGG16, Inception, ResNet, and MDCNN had F1 scores of 93.23%, 92.14%, 94.4%, and 97.56%, respectively. Recall ratings for VGG16 are 91%, Inception is 89%, MDCNN is 96.83%, and ResNet is 92%. The generated MDCNN model performs better than the others based on performance criteria. Figure 5 shows the graphical representation of this comparison. Of these models, the MDCNN exhibits the best performance on all metrics, with the best accuracy and F1 score compared to the typical CNN models assessed.

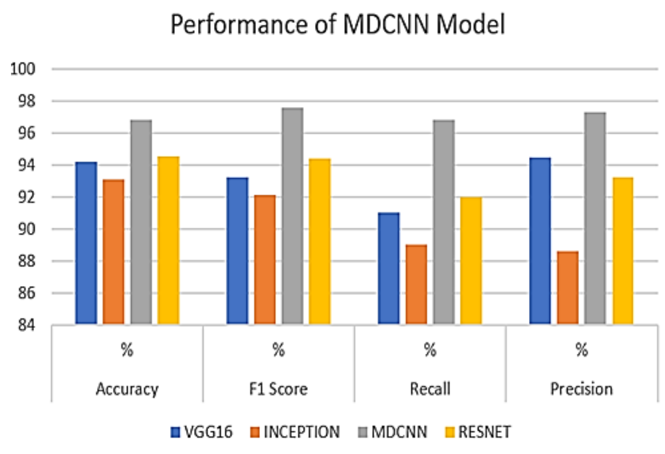

Figure 5 Comparison of Performance Score of MDCNN

This indicates that the MDCNN model effectively balances precision and recall, achieving robust performance in classifying data. The results highlight MDCNN's capability to outperform traditional CNN architectures like VGG16, Inception, and ResNet in tasks requiring detailed classification and recognition tasks. This underscores its suitability for applications demanding high accuracy and reliability in pattern recognition and classification. Figure 5 shows the graphical representation of the proposed model MDCNN performance based on various metrics.

To assess the model's performance in detail, accuracy and loss are key metrics. The proportion of accurately predicted occurrences on the training dataset is known as training accuracy. Elevated training accuracy indicates that the model is interpreting the training data efficiently and gathering up on its patterns. Although a high level of training accuracy is ideal, it does not ensure that the model will function effectively when exposed to new data. Overfitting may occur if the model becomes too specialized in the training data, failing to generalize to new, unseen data. Comparably, validation accuracy calculates the proportion of instances on the validation dataset which is different from the training dataset that is properly predicted. The accuracy ratings of both validation and training are evaluated to show whether or not the model is well-generated or whether over fitting is present. This helps determine the model's effectiveness on new data during the process of execution. Based on this, the hyper parameters can be adjusted appropriately and evaluate the model's efficiency. The dataset's error between real and predicted values is scaled using the loss function. For instance, the categorical cross entropy loss function is mostly used in classification problems. A lower validation loss indicated that the model has performed well on new data, helping to select the best model for training while preventing over fitting and improving its ability to generalize. To reveal how well the chosen model performs on the unknown data, test loss analyses its effectiveness. Evaluating the model's performance in practical situations is helpful. Like validation loss, lower test loss also demonstrates the m odel's efficacy and reliability by showing that the model is p roducing reliable forecasts on unseen data. MDCNN is robust and efficient in analyzing malware in image datasets, as demonstrated by the accompanying diagram displays the model's loss and accuracy score in the diagrammatic representation. The following Figure 6 shows the comparison diagram of validation accuracy and training accuracy to show the efficacy and robustness of the model MDCNN.

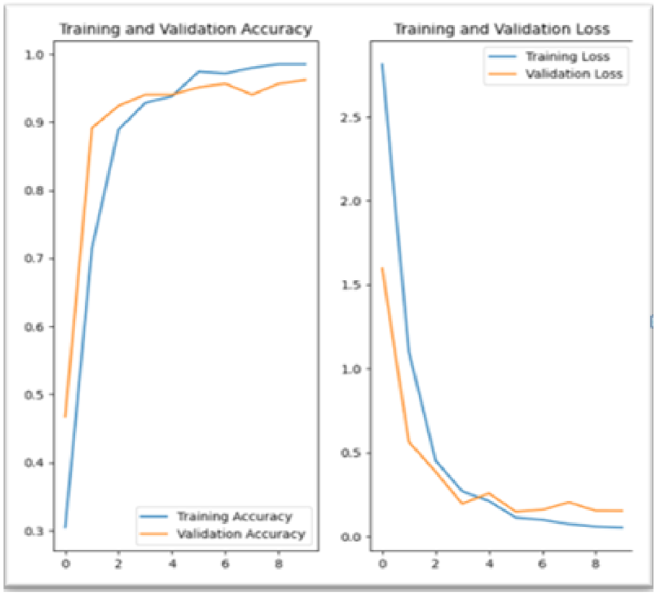

Figure 6 Comparison of Training and Validation Accuracy, Training and Validation Loss

The figure demonstrates that as the number of epochs increases, the accuracy reaches its maximum point, while the loss gradually drops, approaching zero with each epoch. This trend indicates that the model is performing well as it continues to improve and minimize errors during the training process. An important factor in determining a model's efficiency is its optimizer. Every optimizer has distinct qualities that affect the model performance and how it is trained. Concerning this, choosing the best optimizer is challenging but necessary for attaining precise results.

ADAM (Adaptive Moment Estimation)
Momentum and Adaptive learning rates techniques are used by ADAM optimizers to update model parameters. Adapting the learning rates dynamically based on the magnitude and diversity of the gradients, this technique enables swift and accurate convergence. The ADAM optimizer achieves faster convergence and better performance by managing complex and varied data patterns.
ADADELTA (Adaptive Delta) is an optimization algorithm that aims to address some limitations of traditional gradient descent methods by adapting learning rates based on recent gradient information. Using a moving average of the squared gradients, this optimizer continuously alters the learning rate for all the attributes. This approach helps to stabilize and accelerate convergence by reducing the dependence on the initial learning rate and making the optimization process more robust to different scales of parameters. While ADADELTA is effective in various training scenarios, the ADAM optimizer is often preferred for tasks such as malware analysis. ADAM combines the adaptive learning rates and momentum from ADADELTA with additional techniques to enhance convergence and performance, making it particularly suitable for complex and demanding analyses.
Root Mean Square Propagation (RMSprop) is intended to enhance training efficiency for deep learning models. It functions by scaling the learning rate using the mean of squared gradients and modifying the learning rate for each parameter according to the magnitude of recent gradients. This approach helps stabilize and accelerate convergence, especially in scenarios with noisy or sparse gradients. While RMSprop is effective in many training contexts, the ADAM optimizer often provides even better performance, particularly for tasks like malware analysis. ADAM builds on RMSprop by combining its adaptive learning rates with momentum, offering improved outcomes and increased convergence for intricate analysis.
Stochastic Gradient Descent (SGD) modifies the attribute of the model by repeatedly shifting in the direction of the loss function's negative gradient. A small set of data is used to expedite computation and accelerate convergence. While SGD is effective due to its simplicity, the ADAM optimizer often offers superior performance, particularly for malware analysis. ADAM improves upon SGD by incorporating adaptive learning rates and momentum, which dynamically adjust the learning rates and track moving averages of gradients, resulting in more effective convergence and enhanced performance for complex tasks.
Compared to the aforementioned optimizers, our research model MDCNN utilizes the ADAM optimizer due to its adaptive properties and faster convergence rates. ADAM's ability to dynamically adjust learning rates and its effective handling of gradients make it particularly suited for optimizing complex models, leading to more efficient training and improved performance in our malware analysis.

Comparing Training Accuracy and Test Accuracy using Various Optimizers:
Our research compared the accuracy score with various optimizers. Table 2 gives a detailed analysis of accuracy using various optimizers. The AdaGrad optimizer achieves a training accuracy of 96.7% and a test accuracy of 96.6%, indicating its effectiveness in both training and evaluating the model. RMSprop, another optimizer, results in a training accuracy of 95.3% and a slightly lower test accuracy of 94.4%, suggesting it may be less effective in generalizing from the training data compared to AdaGrad. The SGD optimizer shows a training accuracy of 95.24% and a test accuracy of 95.3%, demonstrating consistent performance across both training and testing phases, though slightly lower than AdaGrad and ADAM. In contrast, the ADAM optimizer achieves the highest performance with a training accuracy of 96.83% and a test accuracy of 96.82%.

Table 2 Accuracy-based Comparison of Different Optimizers used in MDCNN Model

Comparison Training Accuracy and Test Accuracy using various Optimizer		
Optimizer	Training Accuracy %	Test Accuracy %
Ada Grad	96.7	96.6
RMS Prop	95.3	94.4
SGD	95.24	95.3
ADAM	96.83	96.82

This consistency in high accuracy across both training and test datasets highlights ADAM's superior capability in effectively training the MDCNN model. Overall, ADAM proves to be the most efficient optimizer among those evaluated, yielding the best results for classifying data with the MDCNN model.

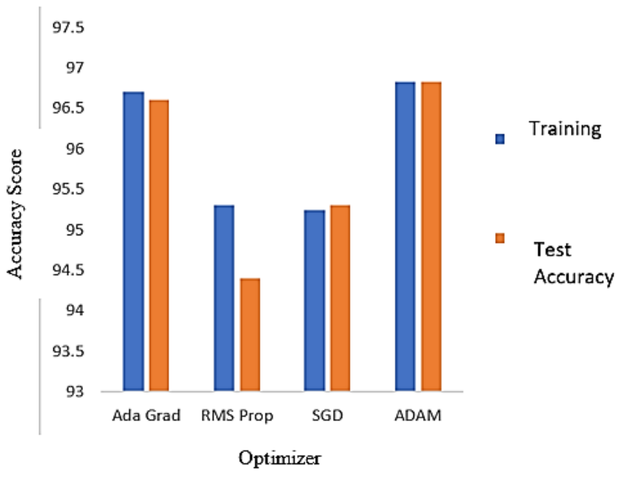

Figure 7 Visual Representation of Accuracy Score Comparison Across Various Optimizers

The bar chart in Figure 7 compares training and test accuracy across four optimizers ADA Grad, RMS Prop, SGD, and Adam. The accuracy score illustrates ADAM optimizer outperforms others.

V Conclusion

In this paper introduces the Malware Detection Convolutional Neural Network (MDCNN). The selected novel approach transforms executable files into images for analysis using image processing and deep learning techniques. This method aims to uncover hidden patterns and signatures that may elude traditional analysis methods. It also discusses experimental results based on an image dataset from VirusTotal, offering insights into the effectiveness of the MDCNN model in enhancing malware detection and response strategies. The MDCNN model achieves a high accuracy rate of 96.83%, surpassing other deep learning models. Comparative performance metrics confirm that MDCNN is exceptionally good at identifying and categorizing malware based on images associated with its family.

REFERENCES

[1] Z. Zhang, G. Fu, F. Di, C. Li, and J. Liu, “Generative Reversible Data Hiding by Image-to-Image Translation via GANs,” Secur. Commun. Networks, vol. 2019, 2019, doi: 10.1155/2019/4932782.

[2] K. A. Zhang, A. Cuesta-Infante, L. Xu, and K. Veeramachaneni, “SteganoGAN: High Capacity Image Steganography with GANs,” 2019, [Online]. Available: http://arxiv.org/abs/1901.03892

[3] R. Zhang, S. Dong, and J. Liu, “Invisible steganography via generative adversarial networks,” Multimed. Tools Appl., vol. 78, no. 7, pp. 8559–8575, 2019, doi: 10.1007/s11042-018-6951-z.

[4] Z. Zhang, G. Fu, J. Liu, and W. Fu, "Generative information hiding method based on adversarial networks," in The 8th International Conference on Computer Engineering and Networks (CENet2018), Springer International Publishing, 2020, pp. 261-270.

[5] Z. Fu, F. Wang, and X. Cheng, “The secure steganography for hiding images via GAN,” Eurasip J. Image Video Process., vol. 2020, no. 1, 2020, doi: 10.1186/s13640-020-00534-2.

[6] D. Vasan, M. Alazab, S. Wassan, B. Safaei, and Q. Zheng, "Image-Based malware classification using ensemble of CNN architectures (IMCEC)," Computers & Security, vol. 92, p. 101748, 2020.

[7] S. Yue and T. Wang, "Imbalanced malware images classification: a CNN based approach," arXiv preprint arXiv:1708.08042, 2017.

[8] J. S. Luo and D. C. T. Lo, "Binary malware image classification using machine learning with local binary pattern," in 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, Dec. 2017, pp. 4664-4667.

[9] S. C. Hsiao, D. Y. Kao, Z. Y. Liu, and R. Tso, "Malware image classification using one-shot learning with siamese networks," Procedia Computer Science, vol. 159, pp. 1863-1871, 2019.

[10] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: visualization and automatic classification," in Proceedings of the 8th International Symposium on Visualization for Cyber Security, July 2011, pp. 1-7.

[11] E. K. Kabanga and C. H. Kim, "Malware images classification using convolutional neural network," Journal of Computer and Communications, vol. 6, no. 1, pp. 153-158, 2017.

[12] A. Azab and M. Khasawneh, "Msic: malware spectrogram image classification," IEEE Access, vol. 8, pp. 102007-102021, 2020.

[13] M. F. Rafique, M. Ali, A. S. Qureshi, A. Khan, and A. M. Mirza, "Malware classification using deep learning based feature extraction and wrapper based feature selection technique," arXiv preprint arXiv:1910.10958, 2019.

[14] M. Sewak, S. K. Sahay, and H. Rathore, "Comparison of deep learning and the classical machine learning algorithm for the malware detection," in 2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), June 2018, pp. 293-296.

[15] F. O. Catak, J. Ahmed, K. Sahinbas, and Z. H. Khand, "Data augmentation based malware detection using convolutional neural networks," PeerJ Computer Science, vol. 7, p. e346, 2021.

[16] J. D. Kothari, "A case study of image classification based on deep learning using TensorFlow," International Journal of Innovative Research in Computer and Communication Engineering, vol. 6, no. 7, pp. 3888-3892, 2018.

[17] F. Ertam and G. Aydın, "Data classification with deep learning using TensorFlow," in 2017 International Conference on Computer Science and Engineering (UBMK), Antalya, Turkey, Oct. 2017, pp. 755-758.

[18] H. Benbrahim, H. Hachimi, and A. Amine, "Deep convolutional neural network with TensorFlow and Keras to classify skin cancer images," Scalable Computing: Practice and Experience, vol. 21, no. 3

[19] P. Kumar and U. Dugal, "TensorFlow based image classification using advanced convolutional neural network," International Journal of Recent Technology and Engineering (IJRTE), vol. 8, no. 6, pp. 994-998, 2020.

[20] S. S. Kadam, A. C. Adamuthe, and A. B. Patil, "CNN model for image classification on MNIST and fashion-MNIST dataset," Journal of Scientific Research, vol. 64, no. 2, pp. 374-384, 2020.

[21] N. Omer, A. H. Samak, A. I. Taloba, and R. M. Abd El-Aziz, "A novel optimized probabilistic neural network approach for intrusion detection and categorization," Alexandria Engineering Journal, vol. 72, pp. 351-361, 2023.

[22] Jose A., J. Arul Valan, and M. Mythily, “Performance Analysis of Mesh based Multicast Routing protocol for Ad-hoc Wireless Networks” International Engineering and Technology Publications Journal of Communication Techniques, Vol. 2, No. 2, pp. 99-103, 2008.

[23] J. Anand, D. Srinath, R. Janarthanan, and C. Uthayakumar, “Efficient Controller Area Network using Multihoming Protocol” Engineering

Today, Journal of Technology World, Vol. XI, Issue 10, pp. 125, -129, Oct. 2009.

[24] Jose A., D. Srinath, R. Janarthanan, and C. Uthayakumar, “Efficient Security for Desktop Data Grid using Fault Resilient Content Distribution” International Journal of Engineering Research and Industrial Applications (IJERIA), Vol. 2, No. VII, pp. 301-313, 2009.

[25] Thanzeem Mohamed Sheriff S., Venkat Kumar J., Vigneshwaran S., Aida Jones, and J. Anand, “Lung Cancer Detection using VGG NET 16 Architecture”, International Conference on Physics and Engineering 2021, IOP Publishing, Journal of Physics Conference Series, Vol. 2040, 2021.