Enhanced Malware Classification: A Hybrid Model Utilizing Denoising Autoencoder and CNN based on visualization method

Thippireddy Harika[1,*], Gera Pradeepini[2]

[1]M. Tech, Dept of CSE, Koneru Lakshmaiah Educational Foundation, Vaddeswaram, Guntur, A.P, India

[2]Professor, Dept CSE, Koneru Lakshmaiah Educational Foundation, Vaddeswaram, Guntur A.P, India

Emails: thippireddyharika@gmail.com; pradeepini_cse@kluniversity.in

Abstract

In the last few years, technology has developed so rapidly that many malware applications are available in the software market. Cybercrimes are increasing day by day with the usage of malware applications. Traditional approaches are not as effective in detecting malware. This study introduces a novel method for distinguishing malware from benign software applications using deep learning models like Denoising Autoencoder and Convolutional Neural Network. Initially, we extract binary code from the applications and transform it into grayscale images. Then, utilizing a denoising autoencoder, we improve the quality of the grayscale images by eliminating noise, and the Convolutional Neural Network uses processed images as input. Finally, the Convolutional Neural Network is employed to differentiate between malicious and benign applications. We test this methodology on the dataset that contains 10,810 malware and 1082 benign files. The suggested model obtains an accuracy of 97% and an F1-score of 96% and performs better than some traditional methods.

Keywords: Cybersecurity; Radare2; Denoising Autoencoder; Convolutional Neural Network; Malware Classification

1. Introduction

Malware is software that cybercriminals create to steal data and damage hardware systems. Some various types of malwares include trojans, worms, spyware, keyloggers, bots, rootkits, and adware [2]. Recent technological breakthroughs have accelerated the growth of mobile internet, which has driven the software sector. The amount of malicious software is increasing rapidly and new malware continues to evolve. Reports show millions of new malwares found and published every day. According to the AV-Test Institute, there are 1,147,501,411-malware software present as of 2024 [3]. Traditional methods often rely on signature-based detection, which matches the malware against a database of known signatures. Deep Learning (DL) is applied in a greater range due to recent advancements in technology and software that have improved accuracy in challenging tasks. Similarly, to enhance cybersecurity systems, Security-related Deep Learning approaches are starting to be used [4].

Numerous methods exist to classify malware. For malware detection, most approaches involve two steps: Steps of pre-processing and classification. The pre-processing phase, Gray-scale images are transformed by extracting DEX files from APKs. Then, the DEX files are changed into Hex files, which are in the format of hexadecimal, decimal, and ASCII. Later, a Markov image is transformed using this Hex file proposed by Dhanya K. A. et al. [5]. Other deep learning methods, which includes the hybrid deep neural network, are also in use, that pre-trains gray-scale images using both AlexNet and ResNet-50 networks, and a feature vector is created by combining the features from Resnet and AlexNet proposed by Ömer Aslan et al. [6]. Mulhem İbrahim et al. [7] designed a static analysis-based technique that extracts all the observable features as opcode sequences, Permissions, broadcast receivers, and services from Android applications. Additionally, they proposed two features, fuzzy hash and file size.

In the classification phase, Dhanya K. A. used a Convolutional Neural Network that contains five layers to classify malware from benign. In addition, this model identifies the type of obfuscated malware. Ömer Aslan used a feature vector obtained from feature extraction using Resnet and AlexNet. For the purpose of classification, this feature

DOI: https://doi.org/10.54216/JCIM.160117
Received: November 14, 2024 Revised: January 14, 2025 Accepted: February 10, 2025

vector is uploaded into the fully connected layer and the softmax layer. Mulhem İbrahim proposed two primary experiments, the first was designed for malware recognition and categorization, utilizing five of the classes in the data set, and the other was developed for malware categorization using dataset classified using two divisions: benign and malware.

In this research, we classify malware from benign applications using Deep Learning techniques. In this approach, we convert Applications to binary code using a versatile reverse engineering framework to convert any malware application to binary code by extracting different features called the Radare2 tool. We convert the binary code to pictures in grayscale. Following that, the images are passed into the Convolutional Neural Network (CNN) and Denoising Autoencoder (DAE) for classification. Our methodology outperforms traditional techniques, according to the experimental results.

This paper's primary contribution is:

1. We suggest a novel approach by extracting binary content from applications. Then read the binary content and convert it into a numpy array of un-signed 8-bit integers.

2. The array is reshaped into matrices and converted into gray-scale images. We use a denoising autoencoder to reduce dimensionality, reduce features, and reconstruct the input images.

3. We propose a convolutional neural network used for classification and evaluate the model.

2. Related Work

Xiaofei Xing et al. [1] suggested a deep learning technique called Autoencoder for malware detection. Benign and malware files are converted into gray-scale images using the Andro guard tool. This paper uses deep learning networks called Autoencoder-1 and Autoencoder-2. Autoencoder-1 is used to analyze whether it is feasible to extract features from gray-scale images or not. Autoencoder-2 is used to detect malware from benign files. The method lacks efficiency in data pre-processing.

Jongkwan Lee et al. [8] suggested a novel technique, which classifies malware based on various Autoencoders using malware images. The malware images are re-sized by using bilinear interpolation. Each Autoencoder is responsible for a specific family of malware i.e., each AE model trains with data from the same family. Before training AE, four hyperparameters must be set: loss function, multiple layers, feature size and numerous neurons per layer. The Proposed model achieves better performance. The limitation is it needs to be updated frequently to maintain its performance.

Yuntao Zhao et al. [9] suggested a technique for detecting malware that combines transfer learning with an enhanced Faster RCNN. They obtained the representative textures by applying a CNN to malware images. A Region Proposal Network produces the target image frame. They pre-trained the Faster RCNN model using ImageNet, a database containing 14 million images. Then, they used the learned approach for classifying malware to improve efficiency. The approach obtained accuracy of 92.8% and a false positive rate (FPR) of 6.8% using 1,821 kinds of malware in five classes that were provided by Microsoft Corporation. The samples were further enhanced by image modification procedures as cropping, flipping and brightness changes.

Iman Almomani et al. [10] developed an automated model based on vision for Android malware detection that consists of 16 efficient and refined CNN algorithms. Color and grayscale images were generated using the byte codes of the "classes.dex" files that were retrieved from Android which were malicious, benign. Then, to effectively and identify Android malware threats, 16 refined CNN algorithms are employed. The accuracy of detection is greater using the suggested model.

Musaad Darwish AlGarni et al. [11] proposed an effective neural network for Transfer Learning-based malware classifications. They proposed CNN pre-trained model learning for carrying out classification on two datasets, Malimg and ImageNet. The model used is Efficient Net B0-B7, which pre-trains on ImageNet that reuses Malimg for malware classification. On many scales, EfficientNet excels at balancing efficiency with precision. This model achieves a high efficiency in classifying malware, but it can be defeated if an adversary understands the technique.

Ömer Aslan et al. suggested a novel deep-learning approach that proposes a hybrid architecture that uses two pre-training networks: AlexNet and ResNet. These pre-trained networks extract both high-level and low-level malware features. Next, the first three fully connected layers show the learning process, and the softMax layer carries out classification. The suggested model yields a 97.78% accuracy rate using three datasets: Malevis, Microsoft Big 2015, and Malimg. The suggested model does not correctly classify certain malware variants that employ sophisticated code obfuscation techniques and share characteristics with other malware kinds. Furthermore, it failed to employ manipulated inputs to test the adversary assaults.

DOI: https://doi.org/10.54216/JCIM.160117
Received: November 14, 2024 Revised: January 14, 2025 Accepted: February 10, 2025

Table 1: Existing Techniques Analysis

S. No	Author	Algorithms Used	Demerits/Future Work	Accuracy
1.	Xiaofei Xing	Androguard Tool, Autoencoder-1 and Autoencoder-2	Data pre-processing method is inefficient and dataset needs to expand.	96.17
2.	Jongkwan Lee	Bilinear Interpolation and Multiple Autoencoders	Frequent updates are needed to maintain its performance.	98.53
3.	Yuntao Zhao	Transfer Learning, Fine-Tuning and Faster Region-Convolutional Neural Networks	Need to explore more datasets.	92.8
4.	Iman Almomani	16 Fine-tuned CNN Algorithms	Difficult to identify different new malware families.	98.05
5.	Musaad Darwish AlGarni	Efficient Net B0-B7 and Convolutional Neural Network	It can be defeated if an adversary understands the technique.	99.93
6.	Ömer Aslan	Pre-trained networks: Alexnet and ResNet, Softmax Layers	It is resistant to obfuscation and cannot test the adversary's attacks.	96.5

3. Proposed Methodology

A. Overview

We propose a new method to classify malicious and benign files that relies on Denoising Autoencoder and Convolutional Neural Networks as shown in Fig.1. Firstly, we decompile applications of malware and benign files and convert them into binary code, then convert it to a NumPy array of unsigned 8-bit integers. Reshape the binary content into square matrices and convert it to gray-scale images. Now, they are loaded into the Denoising Autoencoder, which is used to reconstruct images and remove noise from them. Lastly, perform classification of malware from benign files using Convolutional Neural Networks.

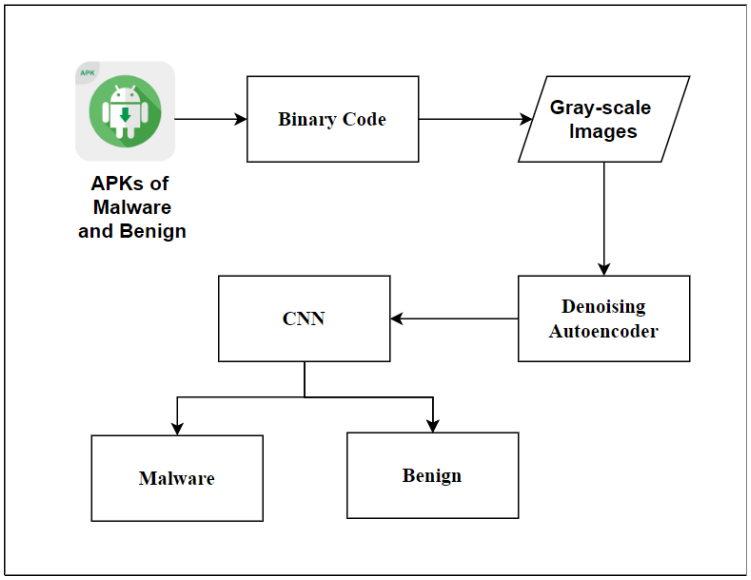

Figure 1. Proposed Methodology

DOI: https://doi.org/10.54216/JCIM.160117
Received: November 14, 2024 Revised: January 14, 2025 Accepted: February 10, 2025

B. Data Preprocessing Phase

The primary aim of the preprocessing stage is providing a neural network an input. The binary code is extracted from benign and malicious applications using the Radare2 (r2) tool. Extracted binary content of files is converted to NumPy array of unsigned 8-bit array. Now, reshape the array into the matrix to form an image of size 64x64. Each value in the array corresponds to a pixel in the gray-scale image. Padding with zero if necessary to fill the image. Gray-scale images are useful for visual analysis and serve as input for neural network models. This visual representation captures the complex details from the malware's byte code that provides a wide range of details for further study. Traditional approaches, namely dynamic and static analysis, requires a lot of time. Therefore, because visualization-based malware detection is more accurate and independent of domain specialists, we prefer it [13]. Since our next step involves neural network models, which need a multi-dimensional fixed size matrix data, the grayscale image that was generated from bytecode of the software may utilized by CNN for training and purposes of classification.

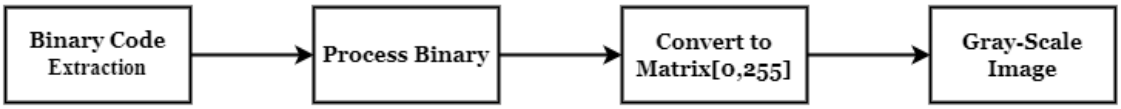

Figure 2. Malware Visualization Process

C. Structure of Denoising Autoencoder

An artificial neural network called a denoising autoencoder is used to learn an image for a given amount of data, usually for unsupervised feature learning or noise reduction. It is an extension of the basic autoencoder, trained to recreate the input data after passing through a lower-dimensional bottleneck. DAE is usually used for training and providing the system with two important aspects. First, DAE preserves the input information, and second, it attempts to remove the noise added to the autoencoder input. Gianni D'Angelo et al.[12] suggested a model to detect malware in androids by using Autoencoders. They used API images as input to sparse autoencoder and used a Soft-max regression-based network for classification.

We designed a structure called DAE as represented in fig 3. That contains an encoder and decoder as shown in Fig. We used DAE to remove noise from gray-scale images to improve features. The given input data is mapped by the encoder phase of an autoencoder to a lower-dimensional image called the bottleneck or latent space. This process compresses a input data, capturing most important features while discarding noise and less relevant information. The encoder's role is to extract the most important information from the data. An autoencoder's decoder role is to reconstructs the input data using the encoder's lower-dimensional image. In a denoising autoencoder, the decoder's purpose is to regenerate the original, noise-free data from the encoded representation. This step is crucial as it enhances the input data's quality, ensuring the most salient features of malware are retained for subsequent classification.

To obtain minimized difference between original data and reconstructed output, the denoising autoencoder is utilized. Mean Squared Error (MSE) is a frequently utilized option for the loss function:

$$L(X,\hat{X}) = \frac{1}{N}\sum_{i=1}^{N}(X_i - \hat{X}_i)^2$$

Where N is number of training samples, X_i is the original input, and $\hat{X}_i$ is reconstructed output.

DOI: https://doi.org/10.54216/JCIM.160117
Received: November 14, 2024 Revised: January 14, 2025 Accepted: February 10, 2025

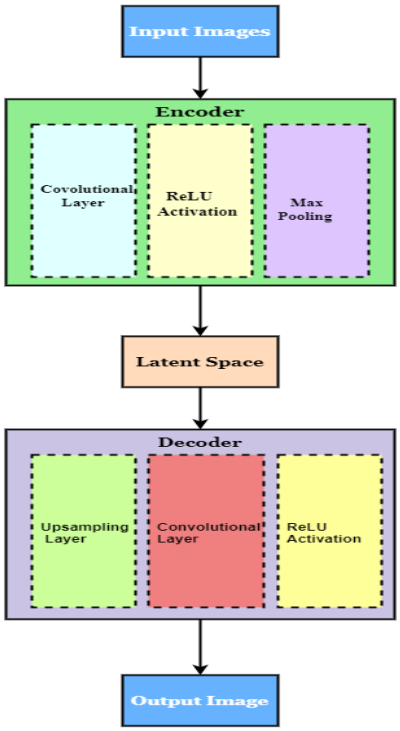

Figure 3. Denoising Autoencoder

D. Classification Phase

For the Classification phase, we use the Convolutional Neural Networks. CNNs are a subset of deep neural networks that are primarily utilized for image analysis. The CNN is made to recognize spatial feature hierarchies from input images efficiently and flexibly. Their capacity to capture temporal and spatial in the data makes them very useful for image classification and identification tasks, lower dimensionality, and fully connected layers carry out the final classification. We use trained encoder part of the autoencoder to encode the training and testing images. To improve the training dataset, use data augmentation techniques like shift, flip and rotation. In order to reduce the classification error, the model's parameters are iteratively modified during the training phase. We use convolutional layers with ReLU activation, max pooling, and dropout for classification. The CNN processes the encoded features through several levels, such as pooling layers to lower dimensionality, convolutional layers to apply filters on the input data, and fully connected layers to carry out the final classification. If malware is detected from the features of the image then it will be labeled as 1, otherwise, 0.

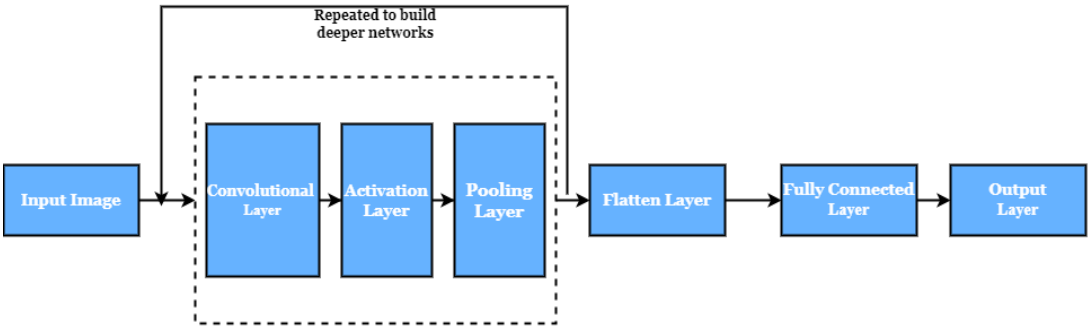

Figure 4. Convolutional Neural Network Architecture

4. Results And Discussion

A. Experimental Setup

We used an Intel CoreTM i5-1135G7 processor with 8GB RAM for our experiments. The operating system on the system was Windows 11 64-bit. For programming, we utilized Python 3.10, TensorFlow 2.15, and Keras.

DOI: https://doi.org/10.54216/JCIM.160117
Received: November 14, 2024 Revised: January 14, 2025 Accepted: February 10, 2025

B. Dataset

The Dataset contains 10,810 malware and 1082 benign files. The malware file includes trojans, malice, spyware, backdoors, etc. The dataset had been separated as 80% for training and 20% for testing. The DAE trains and removes noise from images. The encoded images are trained and classified using CNN.

C. Result Analysis

The comprehensive examination of the DAE is designed to eliminate noise from the input images while maintaining critical features. This process facilitates the generation of a robust and feature-rich representation of the software binaries, which enhances the performance of the CNN classifier. The autoencoder generates denoised variants of the noisy input images as shown in Fig.5. Upon visualizing the images, one would observe that the resultant images exhibit enhanced clarity, with a marked reduction in noise compared to the original input. Through the process of learning to reconstruct images, the DAE identifies significant patterns and features that are useful for classification purpose. This pre-processing phase significantly improves the quality of the input data for the CNN, thereby augmenting the overall classification accuracy.

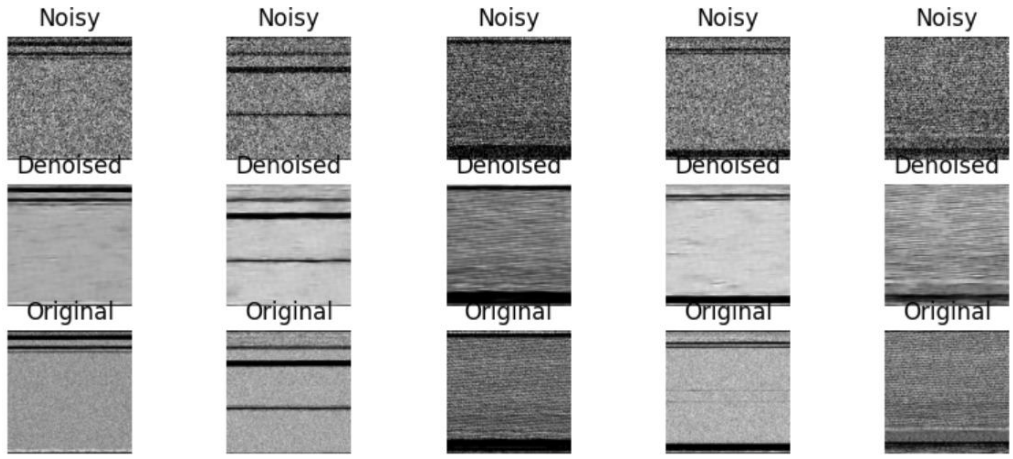

Figure 5. Images of Noisy Input and Denoised Output

We use CNN for classifying malware from benign files. The trained encoded images from DAE are given as input to CNN. The convolutional layers in CNN extract features from the encoded images. The Pooling layers reduce the spatial dimensions that retain significant features. Finally, dense layers learn to classify malware and benign files. The below Fig.6 shows the classification of malware and benign classes. The histogram explains the model's prediction for malware and benign files. The X-axis shows prediction values ranges from 0 to 1, and the Y-axis shows each prediction value's frequency.

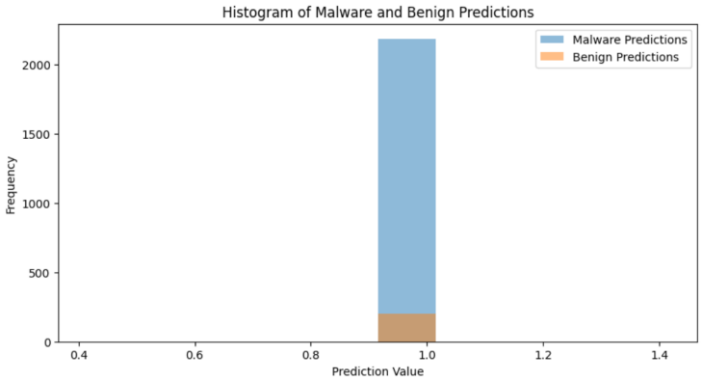

Figure 6. Histogram of Malware and Benign Predictions

D. Evaluation Metrics

Evaluation metrics used in this project to calculate performance are as follows:
1) Precision: Precision indicates the proportion of the model's optimistic estimations that materialize. The difference between the quantity of true-positive and false-positive estimations and the quantity of true-positive predictions is used to calculate it.

DOI: https://doi.org/10.54216/JCIM.160117
Received: November 14, 2024 Revised: January 14, 2025 Accepted: February 10, 2025

2) Recall: The ratio of true positives to the sum of false negatives and true positives is known as recall. It assesses how well the model detects positive samples and is helpful in handling datasets that are unbalanced.

3) F1-Score: A harmonic mean between recall and precision is the F1-Score. It has a range of [0,1]. This statistic often indicates the accuracy and stability of our classifier.

4) Accuracy: The ratio of accurate forecasts to total predictions can be used to calculate accuracy. It provides information about the model's accuracy in making predictions.

$$\text{Precision}=\frac{TP}{TP+FP}$$

$$\text{Recall}=\frac{TP}{TP+FN}$$

$$\text{F1-Score}=2*\frac{\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}}$$

$$\text{Accuracy}=\frac{TP+TN}{TP+FP+FN+TN}$$

Figure 7. Evaluation metrics

True Positive (TP) in this case accurately predicts positive samples. TN is True Negative; it accurately forecasts samples that are negative. False Positive, or FP, predicts positive samples in error. False Negative, or FN, forecasts negative samples in error. Table 1 presents the computed findings.

Table 2: Classification Report

Classification Report	Precision	Recall	F1-Score	Support
Benign	0.67	0.71	0.69	203
Malware	0.98	0.98	0.98	2251
Accuracy			0.97	2395
Macro Avg	0.82	0.84	0.83	2395
Weighted ang	0.97	0.97	0.97	2395

E. Performance Evaluation

The Table 3. shows the performance of existing systems along with our model. Some of the existing techniques are used in our model, but they face some limitations. By overcoming those limitations, our model achieves a high level of accuracy.

Table 3: Comparison Table

S.No	**Methodology**	**Accuracy**
1	Faster RCNN Combining Transfer Learning	92.8%
2	Deep Learning	90%
3	DBN based deep learning	96.76%
4	Parallel CNN Network	94.41%
5	Autoencoder	96.81%
6	Denoising autoencoder & CNN	97.03%

DOI: https://doi.org/10.54216/JCIM.160117
Received: November 14, 2024 Revised: January 14, 2025 Accepted: February 10, 2025

In Fig 8. The accuracy of several models, including our own, is depicted on the graph.

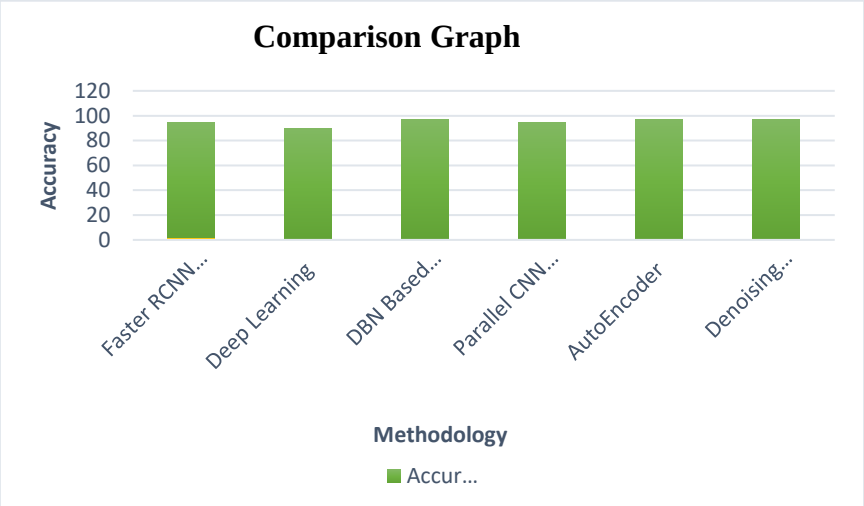

Figure 8. Comparison Graph

6. Conclusion

This Paper suggests a novel method to classify malware from benign files by converting the files into gray-scale images. Our model achieves an accuracy of 97.03% with high Recall, Precision, and F1-score metrics, indicating how effectively it is capable of distinguishing between benign and malicious samples. Compared to traditional methods, our model achieves high accuracy. We are using denoising autoencoders for feature extraction by removing noise and retaining critical features, which leads to better classification performance by CNN. Finally, our model reduces overfitting, improves accuracy, and can be applied to larger datasets. In future work, we will increase the dataset containing unique malware and outline the techniques that should be used to prevent the malware.

Funding: "This research received no external funding"

Conflicts of Interest: "The authors declare no conflict of interest."

References

[1] X. Xing, X. Jin, H. Elahi, H. Jiang, and G. Wang, "A malware detection approach using autoencoder in deep learning," *IEEE Access*, vol. 10, pp. 25696–25706, 2022.

[2] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Comput. Surv. (CSUR)*, vol. 50, no. 3, pp. 1–40, 2017.

[3] AV-ATLAS, "AV-ATLAS Report," 2024. [Online]. Available: https://portal.av-atlas.org/malware?s=1ef56b5f65960fc14ee3dd8f20068779740b96c5&c=eyJrZXkiOiJuZXdNYWx3YXJlUGFzdDE0RGF5cyJ9

[4] E. Rodriguez, B. Otero, N. Gutierrez, and R. Canal, "A survey of deep learning techniques for cybersecurity in mobile networks," *IEEE Commun. Surv. Tutor*, vol. 23, no. 3, pp. 1920–1955, 2021.

[5] K. A. Dhanya et al., "Obfuscated malware detection in IoT Android applications using Markov images and CNN," *IEEE Syst. J.*, vol. 17, no. 2, pp. 2756–2766, 2023.

[6] Ö. Aslan and A. A. Yilmaz, "A new malware classification framework based on deep learning algorithms," *IEEE Access*, vol. 9, pp. 87936–87951, 2021.

[7] M. İbrahim, B. Issa, and M. B. Jasser, "A method for automatic Android malware detection based on static analysis and deep learning," *IEEE Access*, vol. 10, pp. 117334–117352, 2022.

[8] J. Lee and J. Lee, "A classification system for visualized malware based on multiple autoencoder models," *IEEE Access*, vol. 9, pp. 144786–144795, 2021.

DOI: https://doi.org/10.54216/JCIM.160117
Received: November 14, 2024 Revised: January 14, 2025 Accepted: February 10, 2025

[9] Y. Zhao et al., "A malware detection method of code texture visualization based on an improved faster RCNN combining transfer learning," *IEEE Access*, vol. 8, pp. 166630–166641, 2020.

[10] I. Almomani, A. Alkhayer, and W. El-Shafai, "An automated vision-based deep learning model for efficient detection of Android malware attacks," *IEEE Access*, vol. 10, pp. 2700–2720, 2022.

[11] M. D. AlGarni et al., "An efficient convolutional neural network with transfer learning for malware classification," *Wireless Commun. Mobile Comput.*, vol. 2022, no. 1, pp. 4841741, 2022.

[12] G. D'Angelo, M. Ficco, and F. Palmieri, "Malware detection in mobile environments based on autoencoders and API-images," *J. Parallel Distrib. Comput.*, vol. 137, pp. 26–33, 2020.

[13] A. Moawad, A. I. Ebada, and A. M. Al-Zoghby, "A survey on visualization-based malware detection," *J. Cybersecurity*, vol. 4, no. 3, 2022.

DOI: https://doi.org/10.54216/JCIM.160117
Received: November 14, 2024 Revised: January 14, 2025 Accepted: February 10, 2025