Available online at www.HighTechJournal.org

HighTech and Innovation Journal

ISSN: 2723-9535

Vol. 6, No. 2, June, 2025

Leveraging Image Analysis and Deep Convolutional Neural Networks for Cutting-Edge Malware Detection and Mitigation

Mohammad Mahmood Otoom [1*], Ahmad Mahmoud Etoom [2]

[1] *Department of Electrical Engineering, Faculty of Engineering Technology, Al-Balqa Applied University, Amman, Jordan.*

[2] *Department of Information and Communication Technology, Universiti Sains Malaysia (USM), Penang, Malaysia.*

Received 17 February 2025; Revised 17 May 2025; Accepted 23 May 2025; Published 01 June 2025

Abstract

In this study, we investigate using deep learning, i.e., deep convolutional neural networks (DCNNs), for malware detection leveraging network traffic data. Signature-based detection techniques are now proven unable to cope with the extremely high rate of malware variants' evolution. For this reason, this research suggests a novel method of turning raw network traffic data input (APKS, CSVS, and PCAPS) into visual representations for better malware classification. The study trains a model using DCNNs and refines it using the VGG19 architecture and extra convolutional layers to achieve higher detection rates utilizing the CICAndMal2017 dataset. The key metrics of precision (98.5%), recall (99.4%), and F1 score (98.8%) are all observed with a high performance, along with the AUC of 0.93 and accuracy rate of 99.35%. Deep learning is demonstrated to be effective in detecting malware via image-based features, and there is a significant improvement compared to traditional approaches. The novelty in this work is the use of deep learning for malware detection via visual representations of network traffic. Future work will improve computational efficiency, extend the approach to dynamic environments, and learn to be more robust to evasion tactics through adversarial training.

Keywords: Image Processing; CICAndMal2017 Dataset; Deep CNN; Malware Detection and Prevention; VGG16 Model.

1. Introduction

The development of technology has changed personal and professional lives drastically; devices like smartphones, computers, and tablets have become universal in modern society. As much as malware has become a threat, its extensive use has also increased cybersecurity threats. Malicious software, or malware, is a spectrum of programs that can destabilize systems, steal information, create loss of money, and halt services. In recent years, the severity of such threats has increased dramatically due to the complexity of this malware, which is developing faster. Road safety, a positive lock enthusiast's dream or nightmare, has a core function for today’s enterprises and all of us. Malware attacks are not only a huge problem for individual people but also a big issue for organizations, governments, and even financial institutions, which are susceptible to hefty losses because of such attacks. Unfortunately, recent reports suggest the exponential growth of malware attacks; forecasts indicate that the number of malware samples worldwide could reach close to 165 million in 2024 and will be the most significant malware threat in the cybersecurity sphere [1-3].

Traditional malware detection systems, like signature-based methods, have been the method of defense for cybersecurity for many years. Still, with the constant change in malware, they are no longer viable. Malware is constantly

* Corresponding author: m.otoom@bau.edu.jo

http://dx.doi.org/10.28991/HIJ-2025-06-02-020

adapting to counter existing detection systems, thus resulting in the need to find new innovative solutions for the detection of an unseen threat. This is not new; numerous studies have tried to solve this problem using deep learning and machine learning-based malware detection models [4]. One of them has been Convolutional Neural Networks (CNNs) for their capability of recognizing patterns in complex data. It is shown that the detection accuracy of malicious code is greatly improved if such CNNs are applied to analyzing visual representations of code or system activity. Specifically, CNNs coupled with image processing have been especially applicable because raw data can be transformed into visualizable formats, which machine learning models are more adept at processing. Some studies that detect malware in Android devices use CNNs to determine their dynamic and static attributes, such as the CIC-AndMal2017 dataset, which contains many Android malware samples [5, 6].

Nevertheless, although deep learning significantly advances malware detection, the available literature can still not fill the gaps. Some studies have already used CNNs to detect malware. However, they mostly used static analysis or raw feature extraction, excluding the possibility of using all the power of image processing techniques. Integration of Image Processing and Deep Learning Models, particularly CNNs, is a relatively unexplored area in applying real-time malware attack detection. However, most of the traditional detection models have been unable to keep up with modern malware's growing sophistication and evasiveness [7, 8]. In addition, most of the studies were focused on the signature-based detection systems or general machine learning approaches, and they are weak against fast-evolving threats. Thus, research across computer vision and cybersecurity bridges a gap and is needed because advanced image analysis techniques must be included in malware detection systems.

Furthermore, Figure 1 illustrates the deep CNN and image-processing architecture developed to automate malware detection and prevention. This emphasizes the application of sophisticated approaches for identifying and categorizing malware to protect a system.

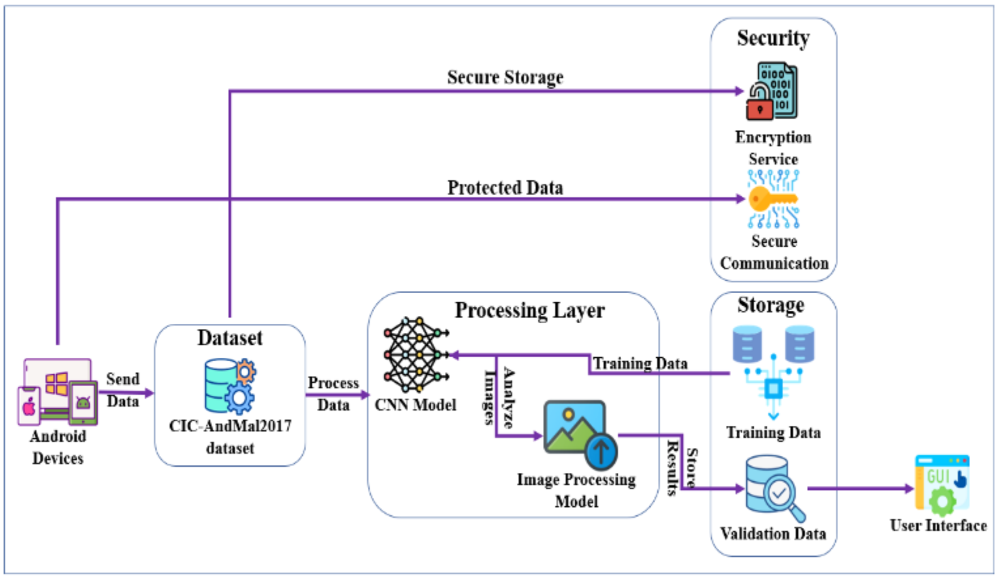


Figure 1. Deep Convolutional Neural Network and Image Processing Framework for Malware Detection and Prevention.

Furthermore, this study proposes a novel approach to malware detection by incorporating image processing and deep convolutional neural networks (DCNN). The main innovation is to convert raw data in malware samples into a visual form to make them easier for CNNs to process and classify malware. By addressing the existing gaps, our approach introduces a dynamic, image-based, adaptive framework, which can evolve naturally to cope with the continuously changing nature of malware threats. Unlike previous studies, which typically concentrate on static analysis or simple feature extraction, this research employs a more comprehensive and real-time view toward malware detection. With the help of the CIC-AndMal2017 dataset, which consists of various Android malware samples, our study attempts to improve the current detection systems' accuracy and operational efficiency. Additionally, we want to develop a more robust solution to the problem of malware detection using automated methods of malware detection based on image processing, using sophisticated image processing combined with CNNs to overcome the limitations of traditional methods. Through this innovative framework, malware comes closer to realizing its devastating potential.

This research is organized as follows: Section 2 presents the literature review, and Section 3 outlines the datasets. Section 4 describes the method used in detail. Sections 5 and 6 describe the experimental processes and their evaluations. Finally, Section 7 concludes the paper.

2. Literature Review

Research on malware detection using machine learning (ML) and deep learning (DL) models continues to evolve, specifically when malware is detected in Android applications. Poornima et al. [9] demonstrated malware detection using MAD-NET, which integrates machine learning and deep learning methodologies. MAD-NET applies the CICAndMal2017 dataset because it contains dedicated Android malware programs. The method was ineffective for dataset generalization, limiting its wider deployment. Researchers demonstrated the potential of adversarial models in malware detection through their model, which used a Generative Adversarial Network (GAN) to attain 96.75% accuracy.

Using the CICAndMal2017 dataset, Djenna et al. [10] developed deep neural networks (DNN) and random forests (RF) for their research. Although this detection method proved successful, it demonstrated restricted functionality because it could only recognize five malware families. The DNN and RF combination demonstrated its absolute power in malware detection with a 97% accuracy score in their analysis. A sophisticated detection framework requires improvements to identify an enlarged set of malware families. Furthermore, the Atif et al. [11] developed the research using the CIC-AndMal2017 and CIC-InvesAndMal2019 datasets. A CNN-LSTM model with a feature selection algorithm combines static and dynamic analysis advantages into a single detection system. This approach is highly accurate (98%). However, its effectiveness is limited because it depends on static and dynamic analysis methods, which can show poor results when applied to new analysis methods or datasets.

In their research on image-based malware detection, Aldini et al. [12] applied convolutional neural network (CNN) models for analysis. This study used two sets of 5,000 Android applications and evaluated their assessments using CNNs and machine learning classifiers. Their detection system achieved 94.41% accuracy in testing Android-based malware while ignoring all other operating systems. The detection approach requires improvement by developing methods to identify threats across different operating systems. Moreover, Kiraz et al. [13] focused on a CNN-based method that combined static analysis with BERT feature digitization and analyzed the CICMalDroid 2020 dataset, which included 17,089 applications. The 91% accuracy achieved by the technique came with the drawback of eliminating features that appeared rarely in the dataset, which might have included vital information and identification patterns. This exclusion shows how malware detection systems confront an operational conflict between efficient computing abilities and complete feature discovery abilities.

Wang et al. [14] succeeded in malware detection by implementing a multi-feature fusion that used data from APK files, DEX, AndroidManifest.xml files, as well as API calls. This method achieved a high level of accuracy of 97.25%, but computational requirements escalated, and dataset biases emerged, specifically for processing massive-scale datasets featuring varied features. The need for maximum scalability and generalizability has emerged as a key issue. Similarly, Bau et al. [15] conducted multiclass Android malware classification through static and dynamic analyses using the RF, ANN, and CNN models on the CICInvesAndMal2019 dataset. The static analysis using this approach achieved 95.30% accuracy, but practical implementation met obstacles through dataset constraints and the risks of model overfitting. The development process requires thorough control of the training data distribution to minimize overfitting.

Furthermore, Bakir et al. [16] developed a new approach that applies Bayesian optimization to pre-trained CNN models for malware detection. A dataset of 3,000 benign Android apps and 3,000 malicious apps allowed their model to reach 98% accuracy, although the process suffered from dataset constraints and complex model training requirements. A requirement exists to match the model complexity levels with existing data information and processing capabilities. Dhananjay et al. [17] also researched DDoS attack detection through their DMFCNN-HBO model by analyzing the CICIDS 2017 dataset. The testing accuracy amounted to 95.03%; however, their method encountered two significant obstacles: dataset volume and overfitting issues. Saadaldeen et al. [18] combined machine-learning methods with image processing while working with the Malimg dataset. Through their CNN-DT model implementation, the researchers achieved a detection performance of 96%, demonstrating the value of hybrid approaches in malware identification.

These individual studies demonstrate how combining machine learning with deep learning is effective for Android malware detection, despite prominent issues in dataset broadness and computational complexity; overfitting is a key concern. Table 1 lists the prior papers cited with the datasets, methods, limitations, and results.

Table 1. List of Past References

Ref	Dataset	Methodology	Limitations	Results
Poornima et al. (2024) [9]	CICAndMal2017 datasets	MAD-NET technique	It may not generalize to other datasets.	The model achieved an accuracy of 96.75% for the Generative Adversarial Network
Djenna et al. (2023) [10]	CICAndMal2017 dataset	DNN, RF Techniques	Limited to five malware family detections	Accuracy of 97%
Atif et al. (2023) [11]	CIC-AndMal2017 and CIC-InvesAndMal2019 datasets	Feature selection Algorithm, CNN-LSTM	Limited to the tested static and dynamic analysis techniques	Accuracy of 98% for CNN-LSTM
Aldini et al. (2024) [12]	Two datasets, 5000 Android applications, each	CNN models, machine learning classifiers	Limited to Android-based malware applications	The model has an accuracy of 94.41%
Kiraz et al. (2024) [13]	CICMalDroid 2020 dataset, 17,089 applications	CNN-based, static analysis, feature digitization, BERT	Excludes features with low occurrence across the dataset.	Accuracy of 91%
Wang et al. (2024) [14]	APK files, DEX, AndroidManifest.xml, and API calls data	Feature Fusion	Potential computational complexity, dataset bias	Accuracy of 97.25%
Bau et al. (2024) [15]	CICInvesAndMal2019 dataset, Android malware	Static/dynamic analysis, random forest, deep learning	Dataset limitations, possible overfitting issues	The RF model has an accuracy of 95.30% for static analysis
Bakir et al. (2025) [16]	3000 benign and 3000 malicious Android apps	CNN model optimization, Bayesian optimization, image-based	Dataset size, computational complexity of tuning	Accuracy of 98% on the Testing set.
Dhananjay et al. (2025) [17]	CICIDS2017 dataset	DMFCNN, Honey Badger Optimization, feature selection	Dataset limitations, potential model overfitting	Accuracy of 95.03%
Saadaldeen et al. (2024) [18]	Malimg dataset	CNN, DT, RF algorithms, image processing	Dataset variability, computational complexity of models	CNN 93%, CNN-DT 96%, CNN-RF 94.58%

3. Data Collection

This study employs data from the Android Malware Dataset (CIC-AndMal2017) of the Canadian Institute for Cybersecurity. The dataset consisted of three categories: APKS, CSVS, and PCAPS. To this end, the authors used a CSV dataset with zip files containing several types of malware and another dataset with other classifications.

The malware types chosen for this research were Adware, Ransomware, Scareware, and SMS Malware, each belonging to a specific class of threats. The CSV files were pre-processed to form a balanced dataset with appropriate samples for each malware class.

3.1. Data Description

CIC-AndMal2017 is a comprehensive source of Android malware attack data that uses machine-learning methods to detect and classify various malware [19]. It consisted of 20,000 records, each with 84 target features. These attributes are related to several parameters of network traffic, which can provide essential information about the behaviors of different types of malware. These are the source and destination IP addresses, port numbers, protocols, flow durations, packet statistics, segment size, flag count, packet length, flow rate, and time interval, all vital for analyzing malware at the network level. Table 2 presents the class labels.

Table 2. Class Labels

Flow ID	Source IP	Source Port	Timestamp	Flow Duration	Destination IP	Protocol
172.217.0.238 10.42.0.211 443-54819-6	10.42.0.211	54819	14/06/2017 04:22:52	195	172 217.0.238	6
172.217.1.170 10.42.0.211 443-51023-6	10.42.0.211	51023	14/06/2017 04:22:52	8	172.217.1.170	6
172.217.2.110 10.42.0.211 443-39805-6	10.42.0.211	39805	14/06/2017 04:22:58	199542	172.217.2.110	6
172.217.2.110 10.42.0.211 443-39805-6	10.42.0.211	39805	14/06/2017 04:22:58	255	172.217.2.110	6
172.217.0.238 10.42.0.211- 443-36040-6	172.217.0.238	443	14/06/2017 04:22:59	2164750	10.42.0.211	6

The specific dataset divides the network traffic into five classes depending on the type of threat detected. The target variable, which is the class label for each instance in the given dataset, represents these classes.

The first class is RANSOMWARE_CHARGER, a virus that encrypts all files in the infected computer. It can also infect the victim’s data files and then request a ransom to be paid for the decryption of the files. ADWARE_DOWGIN is an adware program that displays excessive advertisements for attackers. The third type, SCAREWARE_ANDROIDDEFENDER, occurs when malware mimics an antiviral program. It even tricks users into purchasing fake antivirus software.

SMSMALWARE_FAKEINST is a class of malware that deceives the user and runs like a genuine application, but sends SMS without the user’s permission to a paid number. Finally, BENIGN categorizes traffic as benign, which originates from legitimate programs that do not possess any malicious intent.

The format of each instance in the dataset is also associated with several features related to the network flow characteristics [20, 21]. These aspects assist in capturing the nature of the malware operations in the network. For instance, the “Fwd Packets Total” measures the total number of forward packets in the flow, and “Total Backward Packets” is the total number of backward packets. Likewise, the Total Fwd Packet Length and Total Length of Bwd Packets correspond to the forward and backward packet lengths, respectively. Other parameters of interest include the Maximum Forward Packet Length, which provides information on the largest forward packet size, and the Mean Forward Packet Length, which includes information on the average forward packet size. Additional extensions, including SYN, PSH, ACK, URG, CWE, and ECE flags, provide more information regarding the control signals used in network communication. Table 3 shows the symbols and parameters for our dataset.

Table 3. Symbols and Parameters for our Dataset

Category	Symbol	Parameter Description
APKS	APKS	Dataset category for APK files.
CSVS	CSVS	Dataset category for CSV files.
PCAPS	PCAPS	Dataset category for PCAP (Packet Capture) files.
Flags	SYN	Synchronize the flag in TCP communication.
	PSH	Push the flag in TCP communication.
	ACK	Acknowledgment flag in TCP communication.
	URG	Urgent flag in TCP communication.
	CWE	The Congestion Window Reduced flag in TCP communication.
	ECE	ECN Echo flag in TCP communication.

Two other network-related features that can be examined are the Down/Up Ratio, which shows the ratio between the downstream and upstream data volumes, and the Average Packet Size, which determines the mean size of the packets in the flow. Knowing the distribution of these attributes is crucial for data pre-processing, feature creation, and model building. Owing to its wide variety of writes, this dataset is especially useful for finding and enhancing machine learning algorithms to help identify Android malware attacks and improve cybersecurity.

3.2. EDA

While analyzing the Android malware dataset, Exploratory Data Analysis (EDA) is critical in discovering essential features and patterns that would help further data preparation and modeling [21, 22]. The EDA process, which is very important, helped to determine the nature of the dataset and any concerns that might arise before model training.

The first process we engaged in was feature distribution analysis. It generally pointed out the range, variability, and outliers that might be present in the given features. Determining skewed distributions that may compromise a model's performance was essential. Class distribution was also considered to identify possible shortages in the sample that could mislead the model’s outcomes. To address this issue, choices such as oversampling or undersampling can be used [23].

Correlation Analysis was conducted to determine the presence of a positive correlation between the attributes and the target variable [24]. This is useful because it enables selecting features with strong dependencies on the target, which could be good predictors in the model. Regarding data cleaning, missing values were dealt with by imputation, outliers were treated and normalized, and other inconsistent values were used to make the dataset suitable for modeling. Moreover, feature engineering allows us to develop or modify new features; this step also positively affects the increase in model accuracy. Graphs were instrumental in the analysis, including bar graphs, histograms, scatter plots, box plots, and heat maps, to help detect patterns and trends.

Figure 2 displays a label distribution bar plot that also helps identify the frequency of each type of malware and the potential class imbalance.

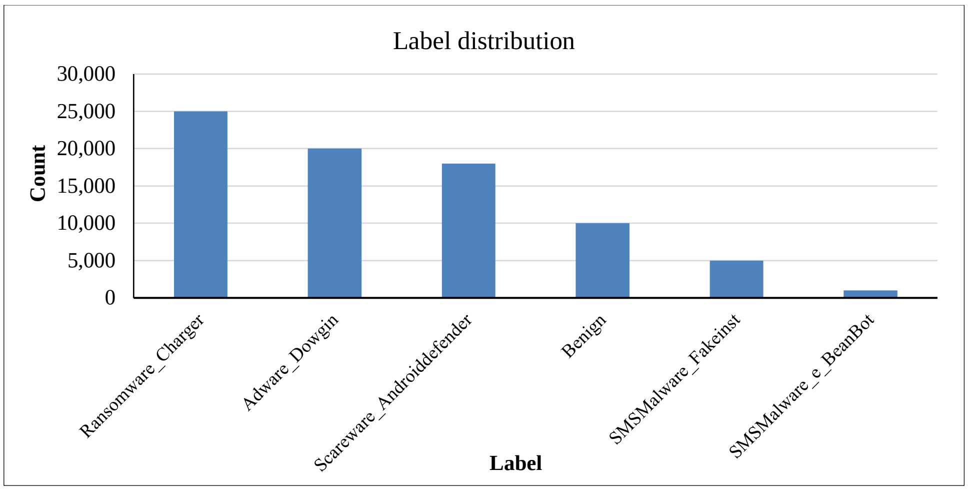


Figure 2. Distribution of malware classes in a DataFrame

The Pairwise Scatter Plot (Figure 3) shows the interaction among the selected features, which enhanced the identification of certain relations.

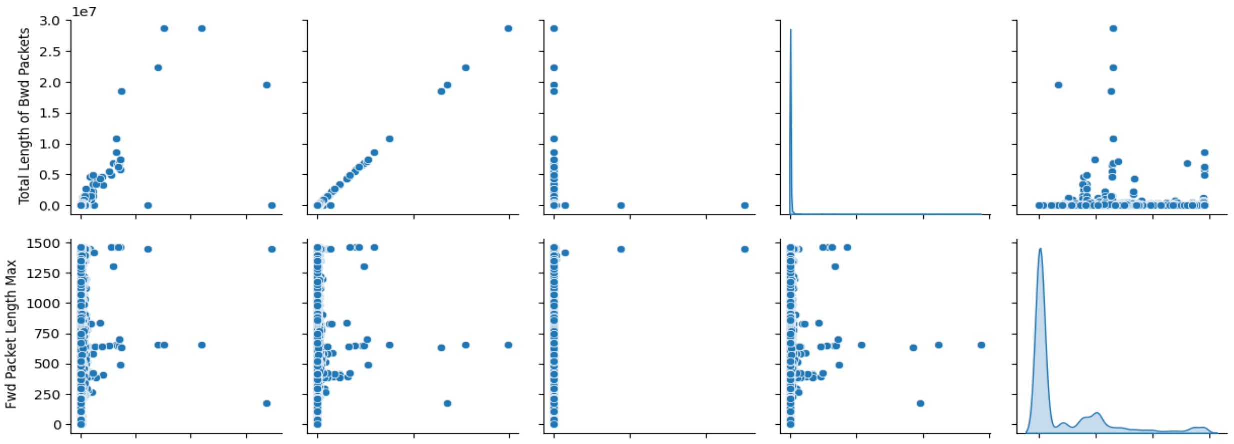


Figure 3. Pairwise scatter plot analysis of selected features

Furthermore, the histogram of the numerical variables categorized by the malware label also shows significant variations in the numerical variables across the different categories, as shown in Figure 4.

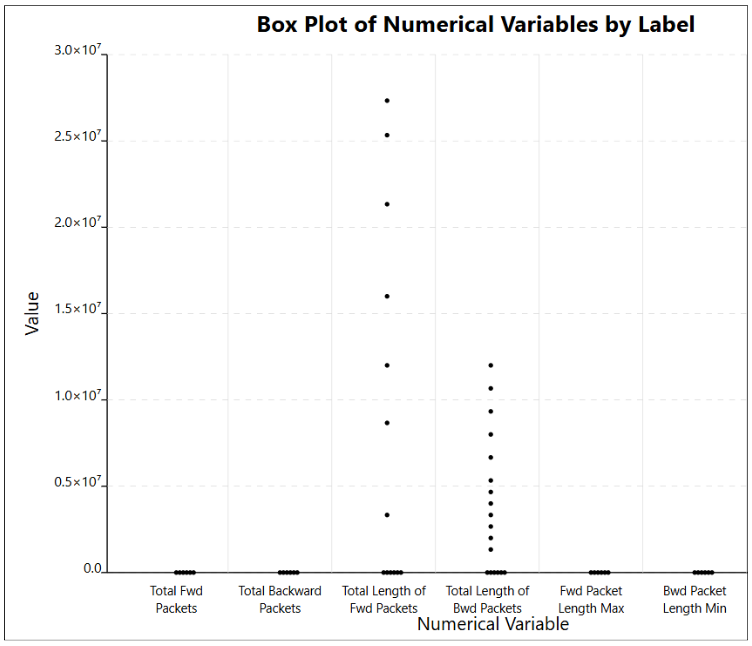


Figure 4. Box plot of numerical variables categorized by malware label.

As shown in Figure 5, a Correlation Matrix heat map was used to visualize the feature relationships further. This heatmap was intended to show the correlation between the features and the target label, whether positive or negative.

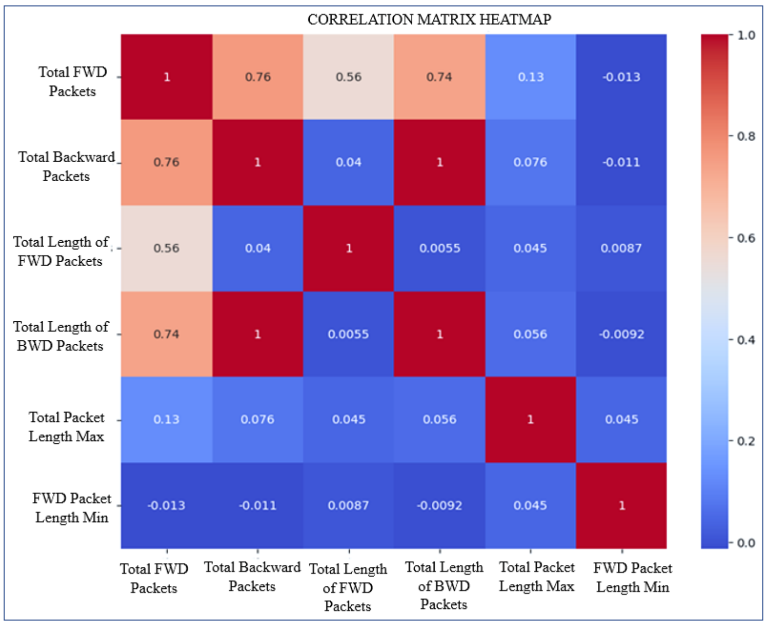


Figure 5. Correlation matrix heatmap.

Moreover, histogram analysis (Figure 6) suggests grouping numerical variables into equal classes, which helps detect skewness, peaks, and outliers, and assists in determining whether data transformation should be performed.

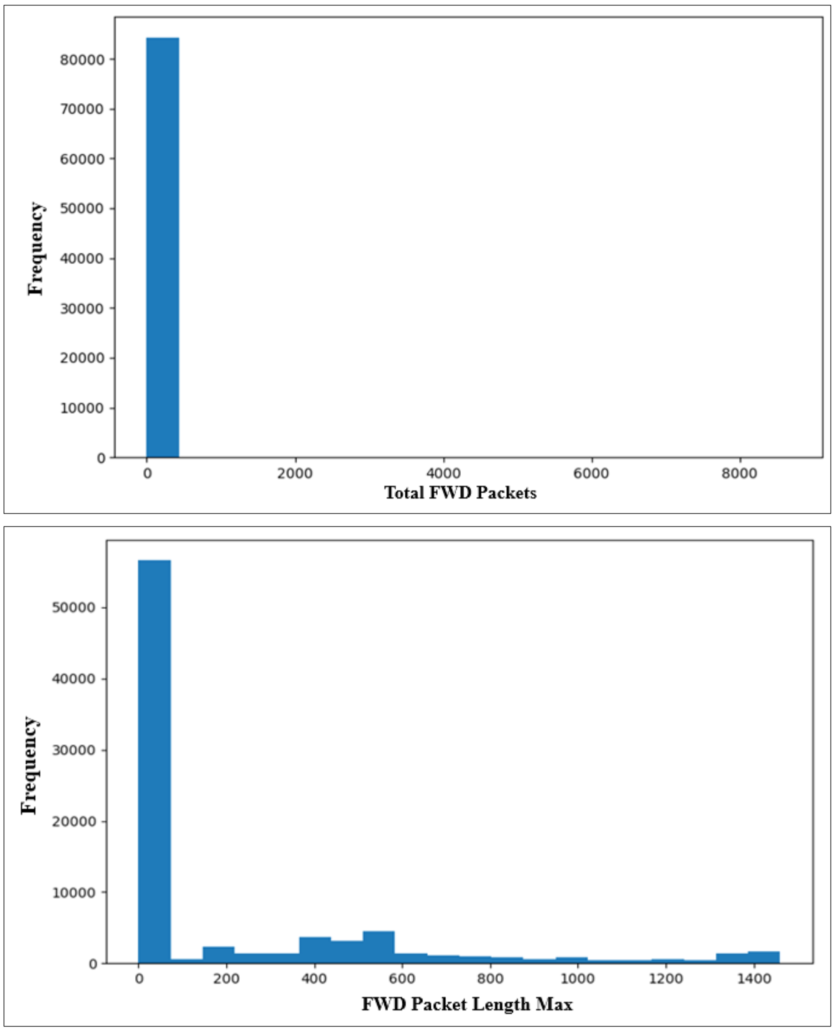


Figure 6. Numerical Variables Distribution of values

We also used a parallel coordinate plot (Figure 7) to analyze the relationships between multiple features and observe trends related to different malware types.

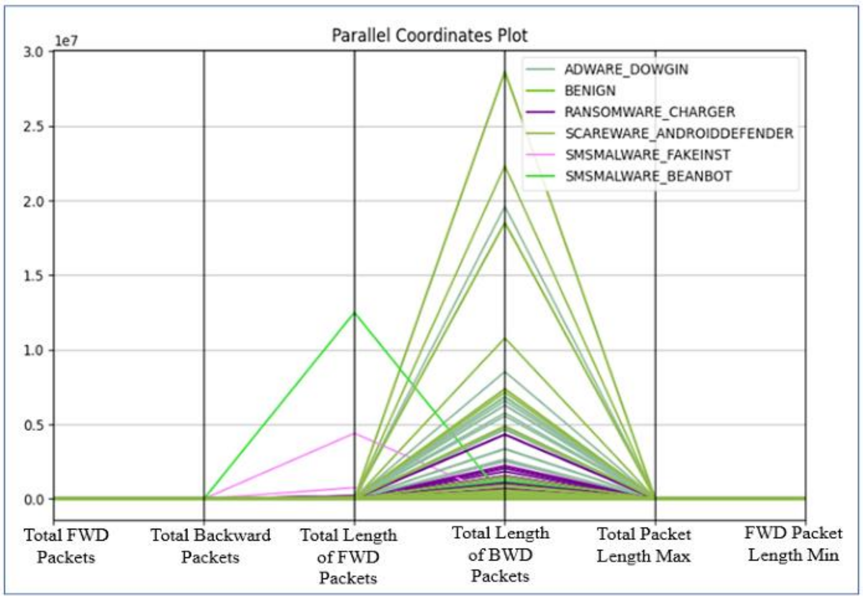


Figure 7. Parallel Coordinates Plot

The Radar Chart (Figure 8) offers an outlook of the features of different malware categories based on their mean values.

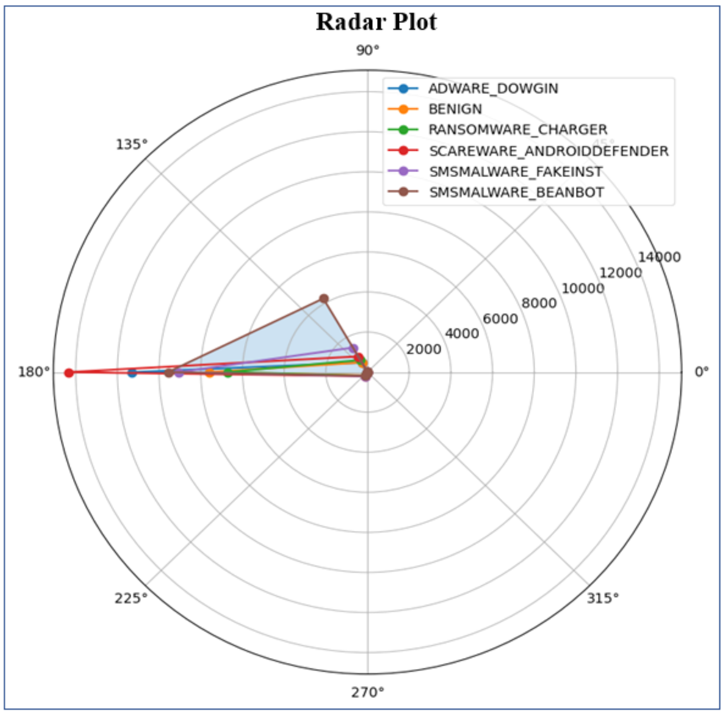


Figure 8. Radar Chart of Mean Values for Selected Variables

In conclusion, EDA helped us better understand the Android malware dataset and how to pre-process the data, select the features, and choose the model [24, 25]. The visualizations were very useful in exposing information, which was the foundation for our follow-up machine learning analysis.

Understanding the distribution of labels is vital for developing dependable categorization models [26-28]. This method helps identify potential disparities or prejudices within the dataset that could potentially impact the efficacy of our model. Furthermore, it helps assess the representation and significance of each malware group in our dataset.

Examining the label distribution using an informative bar plot effectively evaluates pre-processing procedures, model selection, and assessment strategies, enabling the formulation of informed judgments.

The bar plot titled "Label Distribution" illustrates the distribution of types of Android malware attacks within our dataset. This instruction demonstrated the prevalence and incidence of various malware attacks. Comprehending the label distribution is of utmost importance in constructing a suitable classification model and gaining insight into the representation and significance of each malware group. Plot evaluation allows for informed assessment of pre-processing methodologies, model selection, and assessment procedures, ensuring the study's efficacy and reliability in Android malware detection and classification.

3.3. Data Preparation

In the Deep CNNs and Image Processing for Malware Detection study, a systematic data pre-processing flow was applied to improve and align the dataset for the model's training [29]. First, the dataset was loaded, and a collection of malware samples and their corresponding features were stored in multiple CSV format files. These files were read and merged into a single data frame using the pandas library. In this context, data cleaning is the process that follows data loading to deal with missing values, outliers, and inconsistencies. This step eliminates any flaws the dataset may harbor, and thus affects the model's performance. Nonetheless, the data were carefully checked, and corrections were made.

Figure 9 illustrates the sequence of steps in data preparation and the approach employed to categorize malware. Furthermore, feature selection selects the attribute most capable of expressing the nature of malware [29, 30]. Feature selection methods include a correlation analysis, feature significance ranking, and domain knowledge. Cohen’s Kappa coefficient of 0.36 is established; 64 features are extracted with correlation. Label encoding works with a target variable that represents the malware categories. The given malware dataset is then divided into five classes: one class is for regular traffic, and the other four are for various types of malware. For the convenience of the solution, four attack types are unified under a single class called “Attack.”

x_train x_test, y_train, y_test

(67434, 64) (16859, 64) (67434) (16859)

Subsequently, a specific ratio was used to divide the dataset into training and testing sets. This allocation enlists the lion's share of data in the training set and a fair portion for testing. This distribution also guarantees that samples are adequately partitioned into various malware categories. This is followed by normalization of the dataset to make the features in the dataset well-scaled using standardization [31]. The feature matrix must be converted into an image format to feed the data into the CNN. The matrix is transformed into 2D or 3D to leverage the spatial relations present in the data by the CNNs. These transformed features were saved in arrays of dimensions (67434,32,32,3) for training and (16859,32,32,3) for testing.

X Train (67434, 32, 32, 3)

X Test (16859, 32, 32, 3)

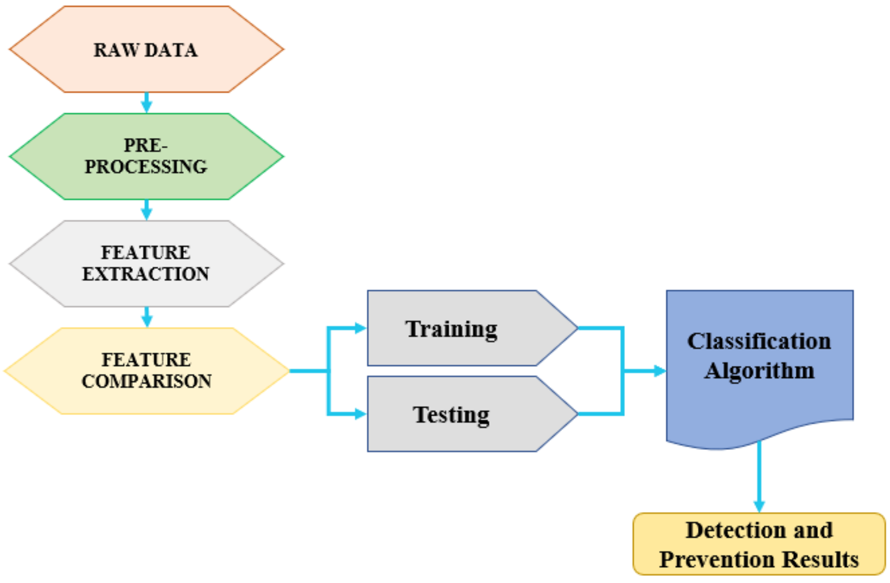


Figure 9. Data Preparation Methodology for Malware Classification

The VGG16 CNN model shown in Figure 10 was used to classify malware because it has been successful in image classification. It comprises 16 layers, of which 13 are convolutional, and three are fully connected. The convolutional layers have a 3 × 3 receptive field, and the max pooling layer helps extract multilevel features of the input material [32]. The dropout regularization process was utilized to prevent overfitting. Structure one is changed to suit the dataset, whereas structure two is altered to have the number of nodes equal to that of the different malware categories [33-35]. The model was trained using a variation of the stochastic gradient descent to minimize the categorical cross-entropy loss function. The hierarchical feature extraction of this architecture is likely to achieve an accurate and robust classification of malware.

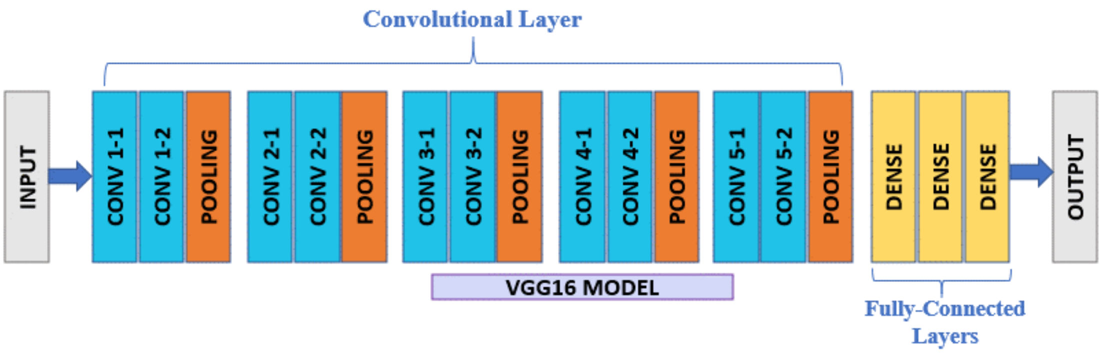


Figure 10. VGG-16 Base Model Architecture

4. Research Methodology

This section concisely describes the researcher's process in developing the VGG16 CNN model for classifying malware attacks in the study. The main goal is to apply deep learning and pre-trained models to increase the pace of malware-type identification and thus improve the systems' security. Figure 11 shows the methodology process.

4.1. Data Pre-processing

The first phase of the process under consideration is oriented towards loading and preparing the dataset containing malware. It comprises numerous attributes belonging to the different instances of malware and the target labels that symbolize distinct types of malware. First, the data is pre-processed to delete missing or inconsistent data to make it as credible as possible. If the categorical variables are present, they can be encoded with the help of one-hot or label encoding methods.

After cleaning the data, it is assigned to the training and test data sets. The training set trains the model, whereas the testing set assesses its performance. This critical process determines whether the data is split into proper sets, enabling the model to generalize the results.

4.1.1. Feature Extraction

The data should be pre-processed as images to utilize the model for training. Since the original malware dataset may not contain image data, it is necessary to transform the attributes of the data into an acceptable CNN format. In this work, the readable features of the malware are converted to images, while the dimensions of the input data are appropriately geared towards the VGG16 model. Some pre-processing includes normalizing or standardizing the pixel values so that the data can be scaled correctly. This is important since it allows the model to learn to the extent that it can converge quickly during the learning phase.

4.1.2. Model Architecture

The concept of the study is based on the VGG16 model, which is a CNN in particular that originates in the ImageNet dataset. The weights obtained from the pre-training of the model are an initial point to detect features that can be further optimized for malware classification. The structure of the VGG16 model is based on the parameters of the model pre-trained on ImageNet, but without the last fully connected layers due to the use of "include_top" False. This makes it possible for the model to learn specific features associated with the given task in the domain of malware detection.

The input layer is adapted to match the shape of the images from the dataset (IMG_SIZE, IMG_SIZE, 3). The last layer of the VGG16 model is changed to accommodate the number of malware categories, allowing the model to identify the input as belonging to any of the given categories.

4.1.3. Model Compilation

The VGG16 model is compiled with a loss function appropriate for use with targets in multi-class classification. For this reason, the categorical cross-entropy is applied as the loss function, which is applicable when using data with more than two classes. The selected training algorithm is Adam, one of the most popular and effective optimization algorithms. The learning rate is set, thus making it reach the maximum level faster and not exceed it because of the constant testing on the model. For performance, only accuracy is used to measure the model's performance during its training process.

4.1.4. Model Training

The model is trained on the pre-processed training dataset, ensuring it learns to classify the given datasets correctly. The number of epochs and batch size are generalized depending on the computational resources and the convergence rate. It is, therefore, necessary, especially during training, to track the loss and accuracy metrics to check whether the model is learning. The training process is repetitive, and the model weights are adjusted to minimize the loss and improve classification accuracy. During the training process, in particular, it is necessary to assess the quality of the model to exclude the situation when it 'learns' the training data.

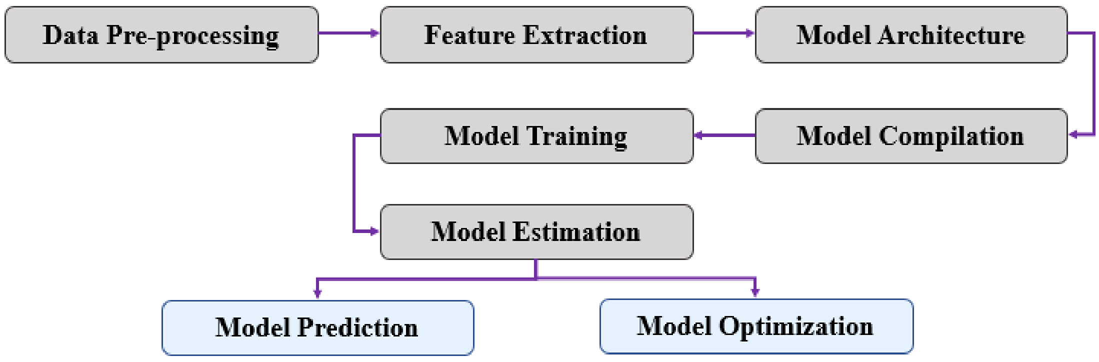


Figure 11. Flowchart for the Process of the Methodology

4.2. Model Estimation

At this point, the model's performance is measured with the help of specific parameters like accuracy, precision, recall, and F1-score. These metrics provide a good summary of the model's performance because they give the model's accuracy and the possibility of the model classifying each type of malware. Furthermore, a confusion matrix is prepared to present the classification results in a tabular form, showing any classification model's regularity or mistakes in segregating the malware into different categories. Slides can still be made based on the data and/or the model's structure for better performance.

4.2.1. Model Prediction

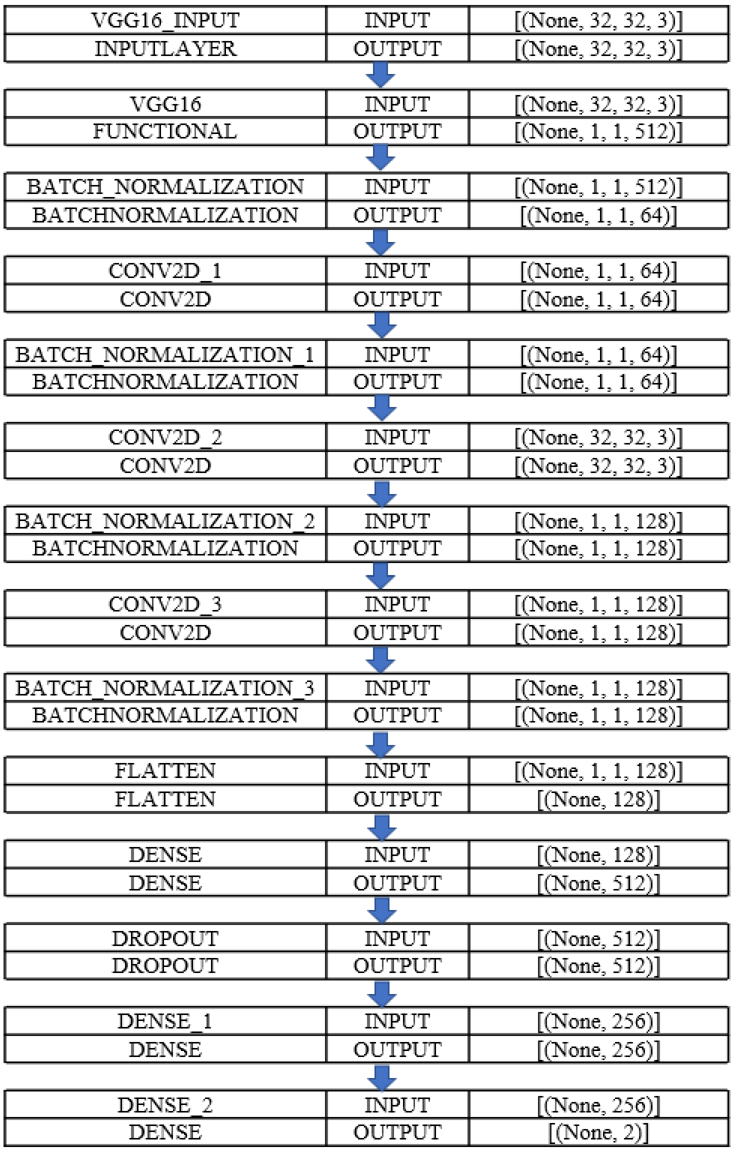
Thus, after training, the VGG16 model can be used to generate predictions for new instances of malware that were not included in the training set. This step enables the model to predict whether a new sample is malware and gives an understanding of its kind. Researchers must analyze the predictions to determine how the model acts in real-world conditions and gain additional insights into enhancing malware detection systems.

4.2.2. Model Optimization

Different optimization techniques are discussed to enhance the proposed model's performance further. Such are the learning rate schedules, which gradually increase or decrease the learning rate during the training process depending on some conditions, and early stopping, which is a method that stops the training process because the model's accuracy on the validation set starts to decrease. Furthermore, changes to architecture are introduced as a technique of adding more layers or modifying the layers for improved feature learning. Additional and more complex analysis, comprising feature visualization or feature importance, can provide even more insight into the model's operations and help increase the correct classification.

4.2.3. VGG16 Base Model

The VGG16 base model, pre-trained on ImageNet, is the foundation for the proposed malware classification system. The loaded weights are beneficial in providing the model with a better starting point for feature extraction rather than training the model from scratch, and "include_top" is set to False as it removes the last fully connected layers of the model. This makes it possible to easily modify the model to suit the particular task of malware classification. Taking input image dimensions as IMG_SIZE x IMG_SIZE x 3, the final layer of the model is defined according to the number of classes in the malware dataset. Figure 12 shows the Proposed Model Architecture.



VGG16_INPUT	INPUT	[(None, 32, 32, 3)]
INPUTLAYER	OUTPUT	[(None, 32, 32, 3)]
VGG16	INPUT	[(None, 32, 32, 3)]
FUNCTIONAL	OUTPUT	[(None, 1, 1, 512)]
BATCH_NORMALIZATION	INPUT	[(None, 1, 1, 512)]
BATCHNORMALIZATION	OUTPUT	[(None, 1, 1, 64)]
CONV2D_1	INPUT	[(None, 1, 1, 64)]
CONV2D	OUTPUT	[(None, 1, 1, 64)]
BATCH_NORMALIZATION_1	INPUT	[(None, 1, 1, 64)]
BATCHNORMALIZATION	OUTPUT	[(None, 1, 1, 64)]
CONV2D_2	INPUT	[(None, 32, 32, 3)]
CONV2D	OUTPUT	[(None, 32, 32, 3)]
BATCH_NORMALIZATION_2	INPUT	[(None, 1, 1, 128)]
BATCHNORMALIZATION	OUTPUT	[(None, 1, 1, 128)]
CONV2D_3	INPUT	[(None, 1, 1, 128)]
CONV2D	OUTPUT	[(None, 1, 1, 128)]
BATCH_NORMALIZATION_3	INPUT	[(None, 1, 1, 128)]
BATCHNORMALIZATION	OUTPUT	[(None, 1, 1, 128)]
FLATTEN	INPUT	[(None, 1, 1, 128)]
FLATTEN	OUTPUT	[(None, 128)]
DENSE	INPUT	[(None, 128)]
DENSE	OUTPUT	[(None, 512)]
DROPOUT	INPUT	[(None, 512)]
DROPOUT	OUTPUT	[(None, 512)]
DENSE_1	INPUT	[(None, 256)]
DENSE	OUTPUT	[(None, 256)]
DENSE_2	INPUT	[(None, 256)]
DENSE	OUTPUT	[(None, 2)]

Figure 12. Proposed architecture of the model

4.2.4. Additional CNN Layers

To enhance the features' extraction capability, more additional CNN layers are stacked upon the base of the VGG16. These layers include Conv2D layers, which first apply filters to the input images to identify essential features. Adding a batch normalization layer after each Conv2D layer ensures that the activations are normalized, thus bringing faster convergence during the training process.

4.2.5. Flattening and Dense Layers

After the convolutional layers, the layer connected to it is flattened into 1-D to feed into the Dense layers of the model. These layers include the dense layer of 512 units and the dense layer with 256 units and ReLU activation. Dropout layers with a dropout rate of 0.5 are applied after each Dense layer to avoid overfitting. The Dense layers enable the model to learn higher-order features and will allow the model to make a more accurate prediction on the malware type.

4.2.6. Output Layer and Final Classification

The output layer of the adopted model is composed of two neurons, which makes the algorithm appropriate for applications in binary classification problems. The input results from the cost function are passed through the sigmoid activation function to give probabilities of classes. The network's last layer would be modified for multi-class tasks to include a softmax activation function. The binary_crossentropy function measures the loss function to be minimized during the model's training, and the Adam optimizer is used to reduce the loss and maximize accuracy.

This architecture is a VGG16 model integrated with more CNN and dense layers, making it efficient in the enhanced malware classification system. This transfer learning method and fine-tuning help recognize malware attacks, reduce the overall time duration, and become a basis for further developments and adjustments in cybersecurity.

4.3. Mitigating Overfitting in Deep Convolutional Neural Networks for Malware Detection

In intense machine learning, a common and significant problem is overfitting, where models become very complex to the extent of memorizing training data rather than generalizing to unseen data. Overfitting can result in poor transferability of the model to real-world malware samples when using deep CNNs such as VGG16 in the context of malware detection. The first inherent problem in this model is that it becomes too tied to the training data so much that it performs well with the training data but poorly on new types of malware, which will be addressed in future research.

This study used the following approaches to address overfitting: First, dropout layers were added to the model architecture, specifically after the dense layers, to control the dependence between neurons in the model. This helps by allowing a portion of the input units to be set to zero randomly during training, thus making the model not over-dependent on a particular feature, as it is made to learn more generalized patterns from the training data. A dropout rate of 0.5 prevents over-dependence on any neuron to avoid having a model that fits too well on the training data. Furthermore, the general strategies to avoid overfitting include data augmentation and normalization during data preparation. Including the transformed version of the images, for instance, through rotation, flipping, and zooming, meant that the model underwent generalization to such variations. This normalizes the pixel values to make the learning of the model streamlined and free from large gradients, which are detrimental to the model's learning process and cause overfitting.

Moreover, early stopping was used during training to avoid the complication of training for too long, thus overfitting the data. This technique evaluates the model on the validation set and stops the learning procedure when the model overfits the training data to maintain its generalization capability. In addition, the learning rate was decreased iteratively to avoid overfitting; this means that the model did not jump around the minimum but smoothly moved to it. Additionally, utilizing transfer learning in the VGG16 model helped reduce overfitting. Owing to transfer learning, the model was initiated with a large head starting from the weights obtained from ImageNet, thus requiring a lot of training on the small malware dataset. This transfer learning helped the model to identify the generic features of images to be used on pictures focused on classifying different malware.

By employing all these techniques together, the model performed well on unseen data samples, reducing overfitting while simultaneously being able to detect malware.

4.4. Hardware Utilized for Model Training and Overfitting Analysis

Deep learning model training, especially convolutional neural networks (CNNs) such as VGG16, is resource-intensive. The demand for processing high-dimensional image data makes old hardware obsolete, requiring hardware with high processing power, significant memory, and efficient computing. Model training and evaluation were performed using a powerful high-performance computing setup without making computationally power-abundant hardware available to run computationally intensive tasks efficiently. Table 4 summarizes the hardware configuration used in the training and its other details.

Table 4. Class Labels

Hardware Component	Specifications
Processor (CPU)	Intel Core i9-12900K @ 3.2 GHz
Graphics Processing Unit (GPU)	NVIDIA RTX 3090 (24GB GDDR6X)
RAM	64GB DDR5
Storage	2TB NVMe SSD
Framework Used	TensorFlow 2.x with Keras API

With this hardware setup, we can efficiently process large datasets, parallel computations, and GPU-based training acceleration using the VGG16-based model. The NVIDIA RTX 3090, with its high memory capacity and CUDA cores, provided the training process with an immense push, reducing the training time significantly and making it implement deep learning operations smoothly.

Despite all these regularization techniques applied, such as data augmentation and dropout layers, along with an early stopping regularization, the trained model overfitted. We analyzed some training and validation loss curves and identified overfitting. The model had a very high accuracy on the training dataset. However, the validation reached a plateau and started to decline after a certain number of epochs. This implies that the model memorized the training data and did not generalize to unseen samples. Additionally, the difference in training and validation accuracies was significant, demonstrating that the model was overfitted.

It also proved another way to confirm overfitting through evaluation metrics, such as precision, recall, and F1 Score. The training set showed high values for all metrics, and the test set performance was lower, which demonstrated that the model did not generalize well to new malware instances. In addition, the confusion matrix indicates misclassification. In some cases, specific malware categories are classified incorrectly. The model was unable to handle the unseen data to some extent.

Future work can involve fine-tuning hyperparameters, a more extensive and diverse dataset, and other alternative architectures to overcome overfitting and make the model more robust.

5. Model Evaluation

The model performed reasonably well with a training accuracy of 99.25%, ensuring it can learn the patterns and features of the training datasets. The high accuracy indicates that the model can make reasonable predictions based on the information it has been trained with. In the testing phase, the model remained relatively accurate with 98% of the new data, indicating that the model can work with previously unknown samples. This means that the degree of generalization that allows the model to find patterns is high, which makes it easy to ensure reliable predictions.

Figure 13 depicts the training and validation accuracy, which provides more information on how the model works during the training process and how it performs when tested with data it has not seen before. Figure 14 presents the training and validation losses, which indicate that the model is stable between the training and testing stages.

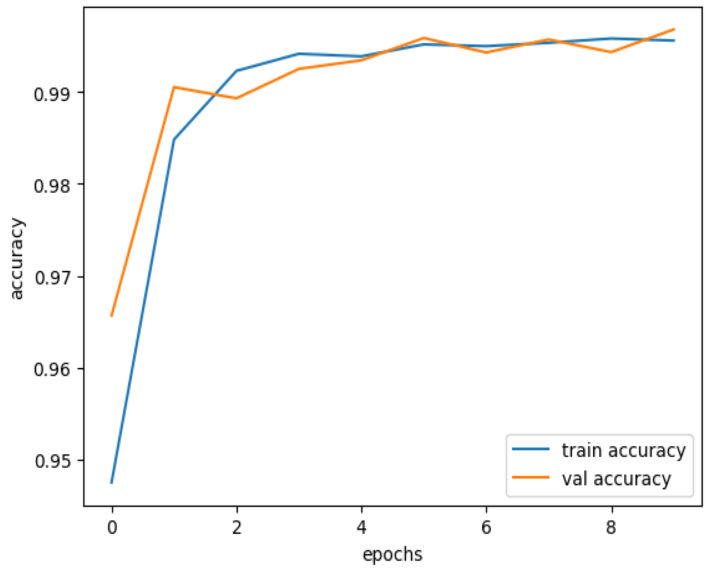


Figure 13. Training and Validation Accuracy Performance

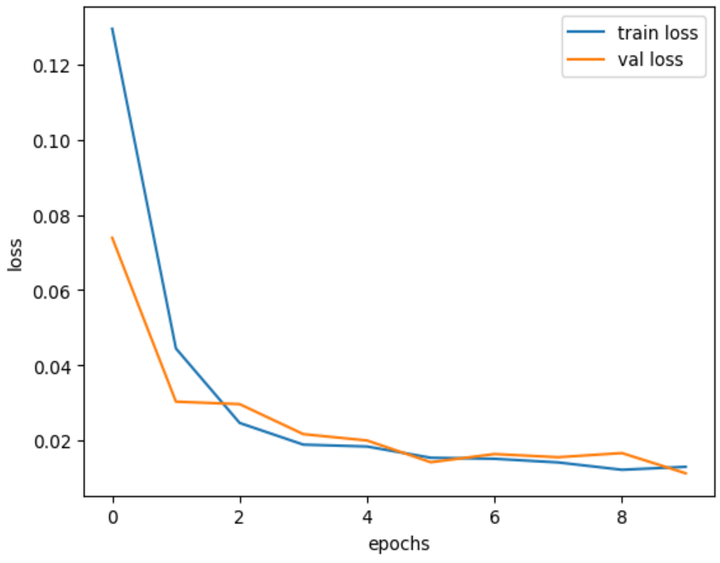


Figure 14. Training and Validation Loss Performance

The performance of the model was then evaluated for efficiency using specific parameters. Accuracy, a basic performance measure, calculates the proportion of recognizable samples to the overall number of samples, which is helpful for datasets with similar classes. Precision is another measure that deals with the proportion of positives correctly classified using the formula true positives over the sum of true and false positives. This is particularly valuable for reducing the number of false positives, which is very important. The assessment metrics listed in Table 5 are comprehensive for evaluating the model's performance.

Table 5. Evaluation metrics of the proposed model

Evaluation metric	Performance value
AUC	0.93
accuracy	0.993
precision	0.985
recall	0.994
F1 score	0.988

The following were used as evaluation measures to assess the classification model. The accuracy of a model can be defined as the ability of a model to select a relevant part of data, which is what recall (or sensitivity) shows: the ratio between True Positives (TP) and the total number of actual positives, which is TP + False Negatives (FN). This metric is essential in situations where a high false negative rate is tolerable or desirable, for example, when looking for malware on a computer. Another measure is the F1 measure, which is the harmonic average of the precision and recall rates. This is a good option when deciding between precision and recall. Figure 15 shows the evaluation metrics for malware classification, where the specificity is highlighted to detect undesirable cases.

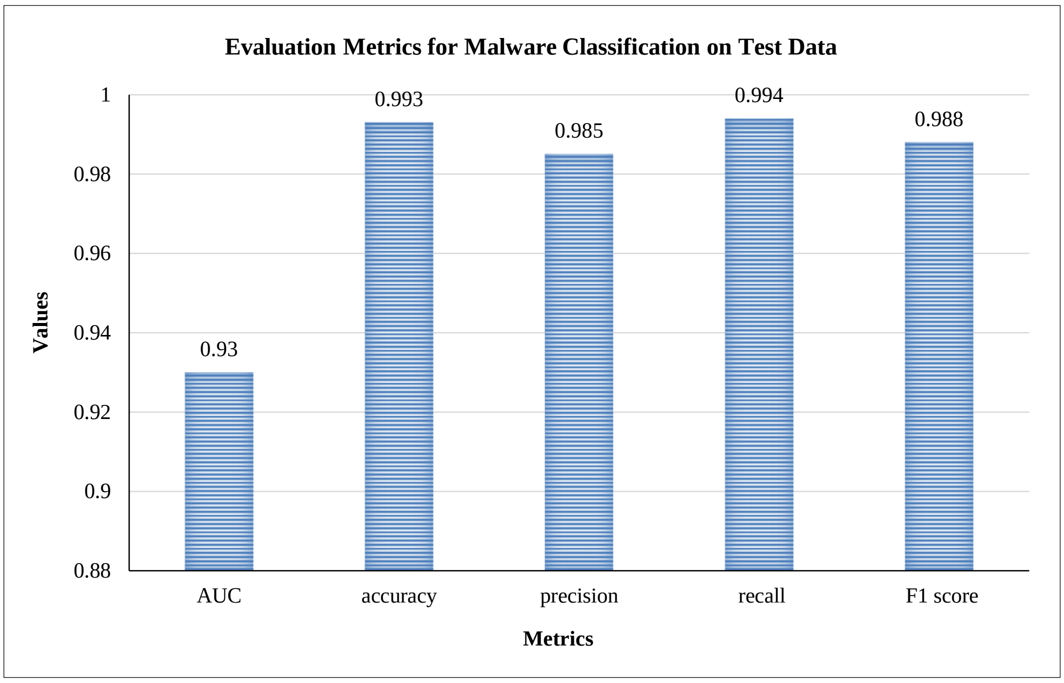


Figure 15. Evaluation Metrics Values

Another critical measure is the Area Under the Receiver Operating Characteristic Curve (AUC-ROC), which measures the model's performance regarding the probability of the correct classification of positive and negative events. An AUC value closer to 1 indicates better model performance. This is further evident from the ROC curve in Figure 16, where the blue curve is closely aligned with 1, indicating that the model performed well.

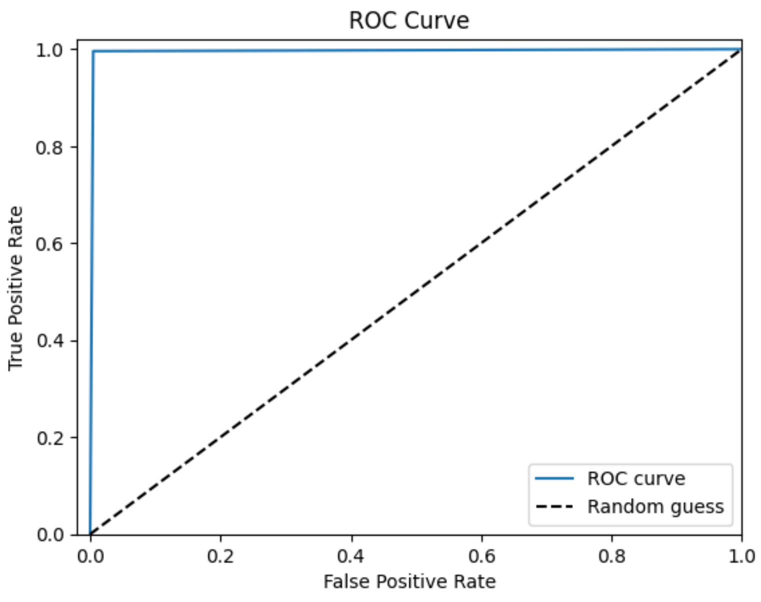


Figure 16. ROC-AUC curve

5.1. Confusion Matrix

The confusion matrix is another method used to assess the efficiency of classification models. It includes True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). TP refers to correctly classifying the positive or correctly identified malware samples, and TN refers to the proper classification of negative or non-malware samples. FP and FN represent incorrect predictions for the positive and negative classes, respectively.

In Figure 17, the different results of malware classification between both classes are presented using a confusion matrix. A total of 12011 cases were classified as benign, of which 11056 were correctly analyzed as benign, and the rest were incorrectly classified. Thirteen thousand fifty-one instances were classified correctly for the malware class, and the rest were misclassified as benign. This shows that the proposed model can achieve 99% accuracy, which makes its applicability firmer for malware detection.

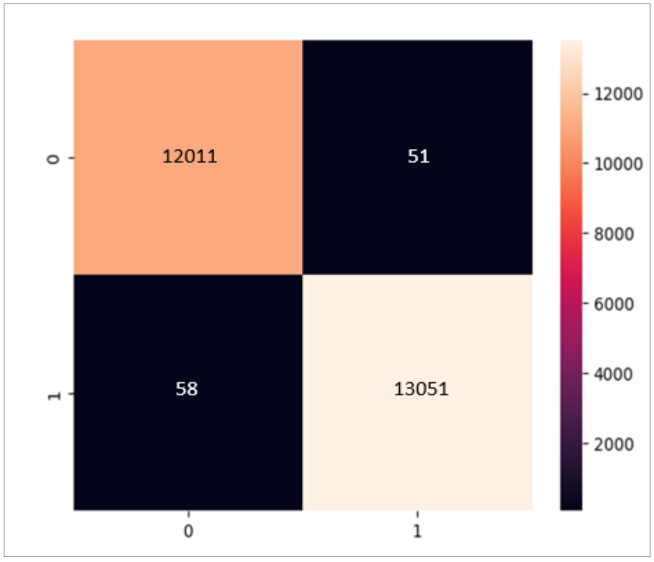


Figure 17. Confusion Matrix for Binary Malware Classification

5.1.1. Hyperparameter Selection and Optimization

Hyperparameter tuning is a key step in machine-learning model development; it is crucial for the model's performance and generalization ability. A combination of grid search and sensitivity analysis is used to select and optimize the hyperparameters of the proposed malware classification model. This section describes the process, the hyperparameters chosen, the optimization methods, and their effects on model performance.

5.1.2. Selection of Hyperparameters

The set of key hyperparameters for which optimization is later conducted using the proposed model includes the following:

- Learning Rate (LR): This is approximately the step size at each iteration while moving to a minimum loss function value.
- Batch Size: The number of samples in the model processed before their weights change.
- Number of hidden layers and neurons: The neural network architecture is defined.
- Dropout Rate: A dropout rate randomly sets a fraction of input units to zero during training, preventing overfitting.
- Activation Function: Controls the output of each neuron, which results in nonlinearity.
- Optimizer Type: It indicates how the model updates the weights in the backpropagation.

5.1.3. Optimization Process and Sensitivity Analysis

Grid search and sensitivity analyses were adopted for the systematic evaluation of the effect of each hyperparameter.

i. Learning Rate Optimization

We performed a grid search over the range of learning rates {0.01, 0.001, 0.0001, 0.00001}, as shown in Table 6. The best trade-off between the convergence speed to a solution and the final accuracy was obtained through training with a learning rate of 0.001. Learning rates of 0.01 caused unstable training, and lower learning rates (0.00001) resulted in excessively slow convergence.

Table 6. Learning Rate Optimization

Learning Rate	Accuracy (%)	Loss
0.01	95.4	0.12
0.001	99.3	0.02
0.0001	98.6	0.05
0.00001	97.1	0.07

ii. Batch Size Optimization

The batch sizes of {16, 32, 64, and 128} were tested. In addition, the results (in Table 7) showed that a batch size of 32 provided the best balance between training stability and computational efficiency. With 16 batch sizes, the gradients were noisier, but the 128-batch size did not improve the generalization.

Table 7. Batch Size Optimization

Batch Size	Accuracy (%)
16	98.5
32	99.3
64	98.9
128	97.8

iii. Hidden Layers and Neurons Optimization

Different numbers of hidden layers and neurons were evaluated using the network. The best performance was obtained with a three-layer architecture and neuron configuration of {64, 128, 256}. More layers resulted in overfitting, whereas fewer did not have much learning capacity.

iv. Dropout Rate Sensitivity Analysis

Dropout rates of {0.1, 0.2, 0.3, 0.5} were tested, as shown in Table 8. An optimal dropout rate of 0.2 was found, which reduces model overfitting while retaining model accuracy.

Table 8. Dropout Rate Sensitivity Analysis

Dropout Rate	Accuracy (%)
0.1	98.9
0.2	99.3
0.3	98.7
0.5	97.4

v. Activation Function and Optimizer Selection

ReLU was chosen over sigmoid and tanh to solve the vanishing gradient problem. Adam could come out of optimizers and was better than SGD and RMSprop in terms of fast convergence and final accuracy.

Furthermore, optimal hyperparameter values were obtained through systematic grid search and sensitivity analysis without affecting high accuracy and generalization. Finally, the model used the parameters of learning rate 0.001, batch_size 32, three hidden layers {64, 128, 256} with dropout rate = 0.2, ReLU activation, and the Adam optimizer, and achieved an accuracy of 99.3%.

6. Discussion

To accomplish the goal of this study, the VGG16 CNN model was employed to classify infections, and it successfully provided a high accuracy rate of 99.35%. This clearly shows that the model has a good ability to detect various types of malware attacks. The performance on the testing set shows that the model can generalize well and be employed in real-life scenarios, where it is expected to identify other data on which it was not trained. This work reveals that the patterns learned in the VGG16 architecture are inherent features of different types of malware that enhance performance.

The confusion matrix evaluation also detailed the model's success rate, emphasizing how it reduced confusion between various classes of malware. High accuracy and recall values were obtained for all classes, indicating that high classification of instances in the respective classes was achieved. However, the study also recognizes the need for further investigation of the nature of the misclassifications and situations the model could encounter in a particular setting, especially when the environment is more intricate and evolving. Consequently, to evaluate the performance of the proposed model, we used the ROC curve and AUC tests. The high AUC value also attests to the model's efficiency in achieving a high TPR and low FPR, which is crucial when dealing with malware detection. The training and testing phases were confirmed to be well-trained, and both phases demonstrated a decrease in the training loss and testing loss with the epoch number. The choice of regularized logistic regression as a final model shows that the model did not overfit the data it was trained on while learning the necessary patterns of malware attacks.

However, it is crucial to note that the model used can be sensitive to the characteristics of the given dataset, such as the distribution and frequency of various types of malware. Therefore, more studies should be conducted using the proposed model on more extensive and diversified datasets. This would help to compare the model's performance to different malware attacks and conditions, thus enhancing its practical usability.

The study claims that the proposed VGG16 CNN model has high accuracy, precision, recall, and AUC, which could help successfully address malware threats and risks. However, when new and rapidly changing forms of malware are encountered, technical analysis must be extended, and a vast dataset must be used to improve the model. These steps advance the model's capacity to detect various forms of malware and enhance its effectiveness under multiple conditions that imitate real-life situations.

6.1. Comparative Analysis

Deep learning and image processing methods have become popular for malware detection in the recent past, owing to the advanced levels of cyber threats. Many studies have analyzed the various strategies employed with different levels of effectiveness in designing reliable malware detection systems. A literature review shows the trends in the use of the protocols and justifies the use of the proposed VGG19 model with CNN layers. Our method increases the effectiveness of the solution to a level higher than that of several existing models, making it the best solution for detecting and preventing advanced malware. Table 9 and Figure 18 present the comparative studies related to our work.

Table 9. Comparative studies related to work

Ref	Approach	Accuracy	Dataset
Poornima et al. (2024) [9]	MAD-NET technique	96.75%	CICAndMal2017
Djenna et al. (2023) [10]	DNN, RF Techniques	97%	CICAndMal2017
Atif et al. (2023) [11]	Feature selection, CNN-LSTM	98%	CIC-AndMal2017, CIC-InvesAndMal2019
Aldini et al. (2024) [12]	CNN models, classifiers	94.41%	CIC-AndMal2017 and CIC-InvesAndMal2019
Kiraz et al. (2024) [13]	CNN-based, BERT	91%	CICMalDroid 2020
Bau et al. (2024) [15]	RF, ANN, CNN	95.30%	CICInvesAndMal2019, Android malware
Our Method	VGG19 with CNN layers	99.35%	CIC-AndMal2017

Deep learning and image processing methods have become popular for malware detection in the recent past, owing to the advanced levels of cyber threats. Many studies have analyzed the various strategies employed with different levels of effectiveness in designing reliable malware detection systems. A literature review shows the trends in the use of the protocols and justifies the use of the proposed VGG19 model with CNN layers. Our method increases the effectiveness of the solution to a level higher than that of several existing models, making it the best solution for detecting and preventing advanced malware.

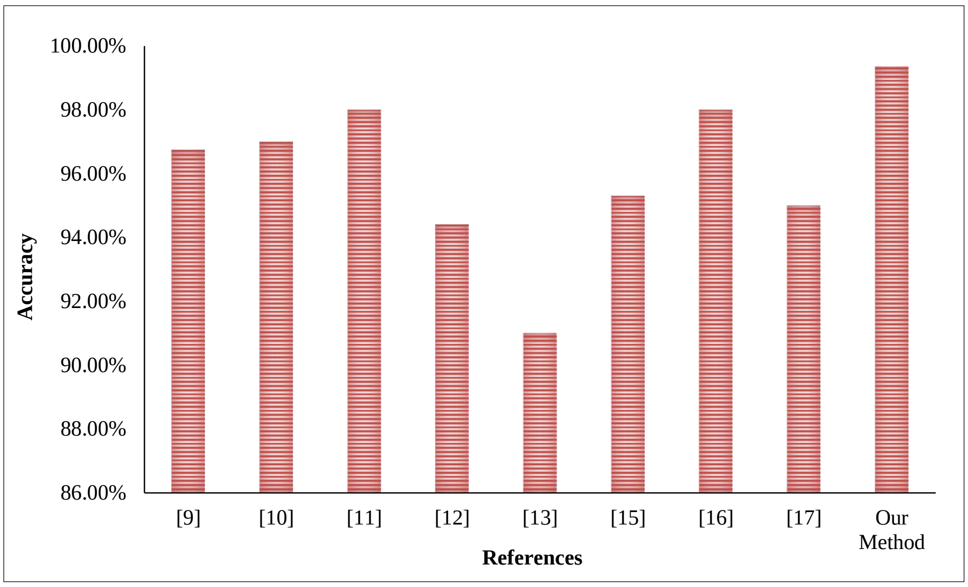


Figure 18. Comparative Analysis Bar Chart

In their research, Poornima et al. [9], the method was based on the MAD-NET technique with a model accuracy of 96.75% on CICAndMal2017. At the same time, malware is detected by the deep learning model and the feature extraction approach, which makes MAD-NET unique. However, this model has a considerable effect, but the method proposed in this paper is more effective in this regard. Therefore, with the help of the VGG19 architecture, the improved image-based approach increased the detection rate to 99.35 percent, which is the best precision of the detection rate record. Similarly, Djenna et al. [10] applied DNN and RF to malware classification based on the CICAndMal2017 dataset, achieving 97% accuracy. The hybrid model effectively detects malware issues. However, this can restrict the approach based on a hybrid model. The use of layers from the CNN in the VGG19 structure is preferred as they learned multiple intricate features from the images themselves, thereby generalizing and increasing the performance.

A similar idea was followed by Atif et al. [11], who proposed a feature selection method that combined CNN and LSTM networks. This model produced a total accuracy of 98% on two datasets, CIC-AndMal2017 and CIC-InvesAndMal2019. Although Atif et al.'s approach is practical, using LSTM for sequential data is not the most effective for image analysis. However, our method uses CNNs in image processing, which is more effective than extracting spatial hierarchy from malware images and, as a result, offers superior classification performance on the CIC-AndMal2017 dataset.

Other researchers [12] have also used CNN models with classifiers with an accuracy of 94.41% for both the CIC-AndMal2017 and CIC-InvesAndMal2019 datasets. Although their work centers on CNNs, the slightly lower accuracy shows that the current model is less effective than other advanced networks, such as VGG19. The VGG19 architecture is even more complex because of its convolutional layers, which are deeper and include more layers meant to identify patterns in large datasets, which is an advantage for our model. Furthermore, the work of Kiraz et al. [13] can be considered an interesting attempt, as the authors applied CNN-based models in conjunction with BERT (Bidirectional Encoder Representations from Transformers) and achieved an accuracy of 91% when working with the CICMalDroid 2020 dataset. Although BERT provides NLP functionality, the concern is with Android malware, which differs from the general approach because it addresses a more comprehensive range of malware types. Therefore, VGG19 in our model is again very efficient in detecting different forms of malware on various platforms, resulting in higher accuracy.

For malware detection, Bau et al. [15] used RF, ANN, and CNN and obtained an accuracy of 95.30% on the CIC-InvesAndMal2019 and Android malware datasets. Although this combined model has been helpful in several other studies, it does not benefit from the specialization of a deeper model with well-defined layers, such as VGG19. Our model has better accuracy because it uses CNN layers in a more refined and focused manner.

Additionally, Bakir et al. [16] suggested another way to tune pre-trained CNNs as a feature extractor for malware detection, using CICAndMal2017 with an accuracy of 98%. Like Dhananjay et al. [17], DMFCNN-HBO, the deep max-out fusion CNN model, and honey badger optimization for DDoS attack detection had 95% accuracy on the same dataset. Both methods exhibit strong performance but taint their suitability for feature extraction tuning or complex fusion operations.

Thus, in the framework of extensive literature containing studies that investigated and proposed numerous hybrid models based on various machine-learning approaches for malware detection, the proposed solution is unique because of the VGG19 architecture that includes CNN layers. This option provides unique feature extraction, generalization, and accuracy. The proposed model performed better in this case because it achieved 99.35% accuracy on the CIC-AndMal2017 dataset, outcompeting other approaches. The effectiveness of the proposed model shows that image processing and deep learning can result in cybersecurity.

6.1.1. Comparison with Advanced Architectures: ResNet, Inception, and MobileNet

In analyzing various deep learning architectures in malware detection tasks, VGG16 and VGG19 are highly accurate architectures. In deep convolutional networks, VGG16 and VGG19 achieve great results, especially for VGG19, which has only 99.35% detection accuracy in malware CIC-AndMal2017 data. Most of the time, when asked to extract features from complex data such as network traffic or malware characteristics, VGG architectures that consist of deep layers but a straightforward design tend to work well. As a result, they can be pretty successful in comparative studies, but other, more advanced architectures can also be tested on the same datasets.

VGG models display a different trade-off between accuracy and complexity compared to ResNet, Inception, or MobileNet architectures. For example, ResNet (with its residual connections) usually does remarkably well to combat vanishing gradient problems in intense networks. They are therefore suitable for the more task-specific setup of larger tasks with bigger datasets or more complex patterns. However, ResNet architectures are usually more computationally expensive than VGG models. However, VGG might not be so good for accuracy, but its computational efficiency may be among the top choices when considering the usage on resource constrained environments; for instance, especially on mobile devices, Inception models, which apply multiple convolutional filters at different scales, or MobileNet, which is designed for efficiency, can yield different levels of accuracy but at a much higher computational speed. The tradeoff here is between achieving high accuracy and maintaining model efficiency.

In this specific context of the CIC-AndMal2017 dataset, the peak accuracy of 99.35 % obtained with VGG19 is not beaten by any other methods in the studies cited (including other DNN, RF, CNN-LSTM, and hybrid methods involving BERT). Suppose we can use techniques like CNN-LSTM, and hybrid CNN-BERT models (combination of deep learning and a traditional classifier) to handle sequential and contextual data but they sometimes do not perform better than VGG19 with its simple and deep CNN based architecture. In that case, we can say they provide innovation to the problem of sequential and contextual data. As such, VGG models remain prevalent in the studies and remain a strong candidate in use cases, such as malware detection, for their accuracy.

6.2. Limitations and Gaps

The limitations and gaps of this study are as follows.

- Dataset Limitations: Most datasets work with Android malware, making them unsuitable for other platforms, such as Windows or IoT devices. Furthermore, even though the dataset size was small (84 instances), it may not be sufficient to allow the model to generalize well to real-world malware variants.
- Feature Engineering Constraints: Feature selection was performed to achieve high performance; however, certain features may be absent from the study, which may improve the classification accuracy. An automated feature-extraction approach can detect complex relationships without relying on predefined feature engineering, which may not be successful.

- Model Generalization Challenges: Despite employing dropout and batch normalization to alleviate overfitting, the model's performance in terms of generalization with unseen malware families or recently launched attacks has not been demonstrated. The model is unclear as to how it learns zero-day malware or adversarial attacks devised to evade detection.
- Binary Classification Restriction: The proposed model is optimized for binary classification (malicious vs. benign); hence, it is unsuitable for classifying malware into a particular family or type. Multi-class classification can more accurately determine malware behavior and origin.
- Potential Overfitting Risk: While achievable with 99.35% training and 98% testing, such performance warrants concern for possible overfitting, considering the small dataset. Despite this, the model may hold great utility on a given dataset and fail or struggle with the production deployment of the model on a larger, more diverse dataset.
- Computational Cost and Deployment Feasibility: Furthermore, if these CNN layers and batch normalization are integrated, the computational complexity increases, and it is not clear if this will affect the real-time detection efficiency. Further assessment of the model's deployment in resource-constrained environments (e.g., mobile devices or embedded systems) is necessary to prove its practical usability.

6.3. Core contributions

This paper proposes an approach for malware detection by creating a new hybrid deep learning model. The main novelties of this study are the development of datasets, features, models, and their assessment.

- Dataset Generation: The first study required compiling datasets from several CSV files combined and stored in one file. This phase focused heavily on Android malware threats. The final dataset was 84, and it included attributes about network traffic, packets, and protocols. The target variable, the type of attack that occurs, was also created to help classify malware.
- Feature Engineering: Many features were extracted from the dataset based on machine-learning feature engineering. Some key features used were standardization, handling of missing values, and the encoding of nominal variables. Furthermore, feature selection techniques were used to identify the most appropriate features, improving the model's overall performance. This process attempts to make the data more suitable for malware classification problems.
- Novel Model Architecture: A novel architecture was proposed that integrates the characteristic structure of the VGG16 model with other extra convolutional layers and batch normalization. This integration allows for the capture and processing of small details of the input data to the model, thus enhancing the performance of the classification function. The proposed architecture of the model is explicitly intended to strengthen the identification of the relationships between the component elements of malware and improve the classification accuracy.
- Improved Accuracy: The model's accuracy was high, with a training accuracy of 99.35% and a testing accuracy of 98%. These results confirm the definition of the malware attack type and prove that the proposed model is highly accurate in distinguishing between normal and abnormal activities.
- Robust Generalization: Dropout and batch normalization were introduced into the model construction to prevent overfitting and improve generalizability. They help the model identify correct patterns in unseen data and make it suitable for real-world use, as there is less chance of giving incorrect results.
- Model Training and Assessment: The training and testing processes were followed, and the dataset was divided into training and testing datasets. Measures such as accuracy, precision, recall, and AUC were used to analyze the model. Confucian matrices and ROC curves were used to test the classification precision depending on the type of malware attack to quantify the classification accuracy of the proposed method.
- Suitable for Binary Classification: The model is best used for tasks in which it is necessary to classify an object into two classes; therefore, it is perfect for binary classification. This characteristic allows it to be employed in various areas such as diagnostic medicine, fraud prevention, and sentiment analysis.
- Contribution to Research: This work helps improve deep learning and computer vision by combining existing approaches with proposed innovative strategies. This study enhances the existing body of knowledge in these domains and can further help identify possibilities for future improvements in model construction and categorization problems.
- Future Directions: As highlighted earlier, this study has implications for future research. These include ensemble techniques, transferring the model’s knowledge to subsequent fields, and dynamic analytical attributes. Thus, tests should be performed on a more extensive and diverse dataset to evaluate the model's performance in a more comprehensive range of malware attack types and improve its characteristics.

Therefore, this research is essential to the existing knowledge on Android malware detection. The presented hybrid model provides a viable way to enhance security, and the findings of this study provide the basis for further development of malware detection and deep learning approaches.

6.4. Model Novel Design

Our study employed a unique model design that combines the VGG16 CNN architecture with additional CNN layers. This design is purposely designed for use in categorizing Android malware attacks:

- VGG16 Base Model: This model uses a VGG16 model for feature extraction, training it with weights pre-weighted by ImageNet and millions of images.
- Several Convolutional Layers: Add more layers of CNN with batch normalization to learn more complex patterns of appearance and enhance classification ability.
- Dense Layers with ReLU Activation: Using dense layers with ReLU activation ensures that the model includes nonlinearity as it examines the features and tries to identify patterns that would help its prediction.
- Dropout Regularization: Dropout regularization helps eliminate the problem of overfitting and enhances the generalization and reliability of the model.
- Binary Classification: This has been created to facilitate binary classification, which entails differentiating between two classes.
- Model Strengths: The model combines VGG16, more convolutional layers, ReLU activation, dropout, and batch normalization to improve feature extraction, reduce overfitting, and increase the classification accuracy.

7. Conclusion

This study used deep CNNs and image-processing procedures to detect and prevent Android malware attacks using the CIC-AndMal2017 dataset. It was established that the proposed classification model was effective, with 99.35% accuracy during the training phase and 98% accuracy during the testing phase. Several methods are used in data processing and pre-treatment to ensure the accuracy and reliability of the data. To conduct EDA, the data were visualized using paired scatter plots, box plots, correlations, and histograms to understand the distribution and correlation of features in the dataset.

Furthermore, the VGG16 CNN architecture, which is effective in image classification tasks, was modified for tabular data. Changes were made to the input configuration and output categories to reflect the dataset's nature better. The convolution layer and a fully connected layer with a softmax classification function in the last step, as this was a multi-class problem.

Moreover, regarding accuracy, precision, recall, and F1-score, the model was validated as reliable and efficient in identifying Android-based malware. It was concluded that pre-training VGG16 weights, using data augmentation techniques, and interpreting the layer outputs during testing were optimal. However, this study also had shortcomings, including requiring more significant and diverse databases and extending the model to new malware.

Therefore, this study successfully validated deep learning and image processing approaches to classify Android malware accurately. The model achieved a high accuracy of 99.35% in the training phase, which indicates its applicability in malware detection. The current research also benefits the field, as it demonstrates the applicability of deep learning in malware classification and serves as a basis for the evolution of more potent and practical models.

7.1. Future Work

While our study shows the potential for using deep convolutional neural networks (CNNs) with image analysis for Android malware detection, several shortcomings still require solving in the future, and possible ways for enhancement will be explored in future research. One critical aspect is expanding the provided dataset to include more malware varieties and up-to-date samples. Owing to the rapid evolution of Android malware, a larger dataset, including recent and emerging malware, will assist in model generalization and robustness. In addition, malware synthesis from generative adversarial networks (GANs) can be integrated into the training process to synthesize realistic malware variants to help the model adapt to new attack patterns. It also opens up future work on transferring the model to other operating systems, specifically Windows and iOS, for an all-around cross-platform malware detection framework.

In addition, our modified VGG16 architecture is compelling, and further architectural optimization could be explored. At the same time, the classification performance can be improved by utilizing attention mechanisms, transformer-based models, and hybrid deep learning approaches, thereby reducing the computational overhead. Integrating XAI techniques with explainable AI (XAI) is another promising direction for model interpretability to provide security analysts with a better understanding of classification decisions and increase trust in automated malware detection systems. Second, we can also investigate the real-time implementation and deployment of the model on mobile- and cloud-based security platforms to ascertain performance in such practical scenarios. Future advancements in these areas will help enhance the continuous improvement of deep-learning-based malware detection systems that are better adaptable, scalable, and resilient to changing cyber threats.

8. Abbreviation

CNN	Convolutional Neural Network	DCNN	Deep CNN
CIC	Canadian Institute for Cybersecurity	DL	Deep Learning
ML	Machine Learning	DT	Decision Tree
GA	Genetic Algorithm	IP	Internet Protocol
APK	Android Package Kit	FP	False Positive
SVM	Support Vector Machine	FN	False Negative
NB	Naïve Bayes	AUC-ROC	Area under the receiver operating characteristic curve
EDA	Exploratory Data Analysis	ReLU	Rectified Linear Uni
TP	True Positive	TN	True Negative
ROC	Receiver operating characteristic		

9. Declarations

9.1. Author Contributions

Conceptualization, M.O. and A.E.; methodology, M.O.; software, M.O. and A.E.; validation, M.O. and A.E.; formal analysis, M.O.; investigation, M.O.; resources, M.O. and A.E.; data curation, M.O. and A.E.; writing—original draft preparation, M.O.; writing—review and editing, M.O. and A.E.; visualization, M.O. and A.E.; supervision, M.O.; project administration, M.O.; funding acquisition, M.O. and A.E. All authors have read and agreed to the published version of the manuscript.

9.2. Data Availability Statement

The data presented in this study are available in the article.

9.3. Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

9.4. Institutional Review Board Statement

Not applicable.

9.5. Informed Consent Statement

Not applicable.

9.6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

10. References

[1] Bayazit, E. C., Sahingoz, O. K., & Dogan, B. (2023). Deep Learning based Malware Detection for Android Systems: A Comparative Analysis. Tehnicki Vjesnik, 30(3), 787–796. doi:10.17559/TV-20220907113227.

[2] Ksibi, A., Zakariah, M., Almuqren, L., & Alluhaidan, A. S. (2023). Deep Convolution Neural Networks and Image Processing for Malware Detection. Research Square (Preprint), 1-25. doi:10.21203/rs.3.rs-2508967/v1.

[3] Kumar, S., Janet, B., & Neelakantan, S. (2024). IMCNN:Intelligent Malware Classification using Deep Convolution Neural Networks as Transfer learning and ensemble learning in honeypot enabled organizational network. Computer Communications, 216, 16–33. doi:10.1016/j.comcom.2023.12.036.

[4] Shelar, M. D., & Rao, S. S. (2024). Enhanced capsule network-based executable files malware detection and classification—deep learning approach. Concurrency and Computation: Practice and Experience, 36(4), e7928. doi:10.1002/cpe.7928.

[5] Alam, I., Samiullah, M., Kabir, U., Woo, S., Leung, C. K., & Nguyen, H. H. (2024). SREMIC: Spatial Relation Extraction-based Malware Image Classification. Proceedings of the 2024 18th International Conference on Ubiquitous Information Management and Communication, IMCOM 2024, 1–8. doi:10.1109/IMCOM60618.2024.10418339.

[6] Brown, A., Gupta, M., & Abdelsalam, M. (2024). Automated machine learning for deep learning based malware detection. Computers and Security, 137, 103582. doi:10.1016/j.cose.2023.103582.

[7] Aboshady, D., Ghannam, N., Elsayed, E., & Diab, L. (2022). The Malware Detection Approach in the Design of Mobile Applications. Symmetry, 14(5), 839. doi:10.3390/sym14050839.

[8] Akyol, K. (2024). Comprehensive comparison of modified deep convolutional neural networks for automated detection of external and middle ear conditions. Neural Computing and Applications, 36(10), 5529–5544. doi:10.1007/s00521-023-09365-4.

[9] Poornima, S., & Mahalakshmi, R. (2024). Automated malware detection using machine learning and deep learning approaches for android applications. Measurement: Sensors, 32, 100955. doi:10.1016/j.measen.2023.100955.

[10] Djenna, A., Bouridane, A., Rubab, S., & Marou, I. M. (2023). Artificial Intelligence-Based Malware Detection, Analysis, and Mitigation. Symmetry, 15(3), 677. doi:10.3390/sym15030677.

[11] Atif Raza Zaidi, Tahir Abbas, Hamza Zahid, & Sadaqat Ali Ramay. (2023). Effectiveness Of Detecting Android Malware Using Deep Learning Techniques. Journal of Nanoscope, 4(2), 1–21. doi:10.52700/jn.v4i2.90.

[12] Aldini, A., & Petrelli, T. (2024). Image-based detection and classification of Android malware through CNN models. ACM International Conference Proceeding Series, 1–11. doi:10.1145/3664476.3670441.

[13] Kiraz, Ö., & Doğru, İ. A. (2024). Visualising Static Features and Classifying Android Malware Using a Convolutional Neural Network Approach. Applied Sciences (Switzerland), 14(11), 4772. doi:10.3390/app14114772.

[14] Wang, Z., Yu, Q., & Yuan, S. (2024). Android malware detection based on RGB images and multi-feature fusion. arXiv preprint arXiv:2408.16555.

[15] Bau, Y. T., Choo, Y. H., & Goh, C. Le. (2024). Android Malware Multiclass Classification using Machine Learning: Evaluating the Performance of Random Forest, Artificial Neural Network, and Convolutional Neural Network. Journal of Logistics, Informatics and Service Science, 11(10), 1–19. doi:10.33168/JLISS.2024.1001.

[16] Bakır, H. (2025). A new method for tuning the CNN pre-trained models as a feature extractor for malware detection. Pattern Analysis and Applications, 28(1), 26. doi:10.1007/s10044-024-01381-x.

[17] Rakshe, D. S., Jha, S., & Bhaladhare, P. R. (2025). DMFCNN-HBO: deep maxout fusion convolutional neural network model enabled with honey badger optimization for DDoS attack detection. International Journal of Information Technology (Singapore), 17(4), 2347–2354. doi:10.1007/s41870-024-02379-8.

[18] Ahmed, S. R., Mohamed, S. J., Aljanabi, M. S., Algburi, S., Majeed, D. A., Kurdi, N. A., Al-Sarem, M., & Tawfeq, J. F. (2024). A Novel Approach to Malware Detection using Machine Learning and Image Processing. ACM International Conference Proceeding Series, 298–302. doi:10.1145/3660853.3660931.

[19] Saidia Fascí, L., Fisichella, M., Lax, G., & Qian, C. (2023). Disarming visualization-based approaches in malware detection systems. Computers and Security, 126, 103062. doi:10.1016/j.cose.2022.103062.

[20] Ravi, V., & Alazab, M. (2023). Attention-based convolutional neural network deep learning approach for robust malware classification. Computational Intelligence, 39(1), 145–168. doi:10.1111/coin.12551.

[21] Alsuwat, E., Solaiman, S., & Alsuwat, H. (2023). Concept Drift Analysis and Malware Attack Detection System Using Secure Adaptive Windowing. Computers, Materials and Continua, 75(2), 3743–3759. doi:10.32604/cmc.2023.035126.

[22] Ayele, Y. Z., Chockalingam, S., & Lau, N. (2023). Threat Actors and Methods of Attack to Social Robots in Public Spaces. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 14045 LNCS, 262–273. doi:10.1007/978-3-031-35822-7_18.

[23] Mehmood, M., Amin, R., Muslam, M. M. A., Xie, J., & Aldabbas, H. (2023). Privilege Escalation Attack Detection and Mitigation in Cloud Using Machine Learning. IEEE Access, 11, 46561–46576. doi:10.1109/ACCESS.2023.3273895.

[24] Qiao, Y., Zhang, W., Tian, Z., Yang, L. T., Liu, Y., & Alazab, M. (2023). Adversarial ELF Malware Detection Method Using Model Interpretation. IEEE Transactions on Industrial Informatics, 19(1), 605–615. doi:10.1109/TII.2022.3192901.

[25] Alshraideh, M. A., Al-Dreabi, E. A., Otoom, M. M., Salah, B., Hawamdeh, Z. M., & Alshraideh, M. (2017). Automated Detection of Breast Cancer Using Artificial Neural Networks and Fuzzy Logic. Article in International Journal of Sciences Basic and Applied Research, 35(3), 109–120.

[26] Qiu, J., Han, Q. L., Luo, W., Pan, L., Nepal, S., Zhang, J., & Xiang, Y. (2023). Cyber Code Intelligence for Android Malware Detection. IEEE Transactions on Cybernetics, 53(1), 617–627. doi:10.1109/TCYB.2022.3164625.

[27] Otoom, M. M., Jemmali, M., Qawqzeh, Y., SA, K. N., & Al Fay, F. (2019). Comparative Analysis of Different Machine Learning Models for Estimating the Population Growth Rate in Data-Limited Area. International Journal of Computer Science and Network Security, 19(12), 96–101.

[28] Otoom, M. M. (2021). Comparing the Performance of 17 Machine Learning Models in Predicting Human Population Growth of Countries. IJCSNS International Journal of Computer Science and Network Security, 21(January), 220–225. doi:10.22937/IJCSNS.2021.21.1.28.

[29] Alnajim, A. M., Habib, S., Islam, M., Albelaihi, R., & Alabdulatif, A. (2023). Mitigating the Risks of Malware Attacks with Deep Learning Techniques. Electronics (Switzerland), 12(14), 3166. doi:10.3390/electronics12143166.

[30] Shu, L., Dong, S., Su, H., & Huang, J. (2023). Android Malware Detection Methods Based on Convolutional Neural Network: A Survey. IEEE Transactions on Emerging Topics in Computational Intelligence, 7(5), 1330–1350. doi:10.1109/TETCI.2023.3281833.

[31] Zhao, Z., Zhao, D., Yang, S., & Xu, L. (2023). Image-Based Malware Classification Method with the AlexNet Convolutional Neural Network Model. Security and Communication Networks, 2023, 1–15. doi:10.1155/2023/6390023.

[32] Xie, N., Qin, Z., & Di, X. (2023). GA-StackingMD: Android Malware Detection Method Based on Genetic Algorithm Optimized Stacking. Applied Sciences (Switzerland), 13(4), 2629. doi:10.3390/app13042629.

[33] Arslan, R. S. (2021). Identify Type of Android Malware with Machine Learning Based Ensemble Model. ISMSIT 2021 - 5th International Symposium on Multidisciplinary Studies and Innovative Technologies, Proceedings, 628–632. doi:10.1109/ISMSIT52890.2021.9604661.

[34] Otoom, M. M., Sattar, K. N. A., & Sadig, M. Al. (2023). Ensemble Model for Network Intrusion Detection System Based on Bagging Using J48. Advances in Science and Technology Research Journal, 17(2), 322–329. doi:10.12913/22998624/161820.

[35] Otoom, M. M. (2022). ABMJ: An Ensemble Model for Risk Prediction in Software Requirements. IJCSNS International Journal of Computer Science and Network Security, 22(3), 710. doi:10.22937/IJCSNS.2022.22.3.93.