Journal of Computing & Biomedical Informatics
ISSN: 2710 - 1606

Volume 08 Issue 01
2024

Research Article
https://doi.org/10.56979/801/2024

Dynamic Malware Detection in Wireless Networks using Deep Learning

Muhammad Aitzaz Ahsan[1], Muhammad Munwar Iqbal[1], Habib Akbar[1], Shaban Ramzan[2], Hamza Badi Uz Zaman Khan[1], Umair Khadam[3], and Muhammad Umar Chaudhry[4*]

[1]Department of Computer Science, University of Engineering and Technology, Taxila, 47080, Pakistan.
[2]Department of Computer Science & IT, Government Sadiq College Women University, Bahawalpur, Pakistan.
[3]Department of Software Engineering, University of Kotli, AJK.
[4]Department of Computer Engineering, Bahauddin Zakariya University, Multan, Pakistan.
[*]Corresponding Author: Muhammad Umar Chaudhry. Email: umar.chaudhry@bzu.edu.pk

Received: June 02, 2024 **Accepted:** September 28, 2024

Abstract: In the current era of fast digital growth, the significance of security cannot be emphasized enough. Many academics have focused their efforts on creating malware detection systems that utilize data mining techniques to monitor and detect any security breaches. Nevertheless, despite these technological developments, existing systems continue to face challenges in attaining the necessary degree of precision for exact detection. Modern malware employs various evasive techniques, such as polymorphism and metamorphism, to rapidly change and generate numerous variants, challenging traditional detection methods. While machine learning algorithms (MLAs) have shown promise in malware analysis, they often suffer from slow performance due to extensive feature engineering and representation requirements. Advanced deep learning models can eliminate the need for feature engineering but may still face issues with biased performance due to skewed training data, which limits their real-time applicability. This research addresses these challenges by evaluating both classical MLAs and deep learning architectures for malware detection, classification, and categorization. Using a diverse set of public and private datasets, we performed experimental analyses with various dataset splits to train and test models over different timescales. Our key contribution is the development of a novel image processing technique with optimized parameters for MLAs and deep learning models, aimed at improving the effectiveness of zero-day malware detection.

Keywords: Malware; Wireless Networks; Deep Learning; Machine Learning Algorithms (MLAs).

1. Introduction

Wireless technologies, mobile devices, and networks have made it easier to process a lot of data. However, such improvements introduce serious security weaknesses, making systems open to a variety of threats and malicious attacks. Wireless communications are open, flexible, and portable, which exacerbates security threats. To counteract these risks, intrusion detection systems (IDS), both host and network, are critical in safeguarding these networks [1]. An effective intrusion detection system (IDS) must be efficient, robust, and capable of reliably detecting threats while limiting false positives and handling alert frequency. Recent research is heading toward employing machine learning to increase IDS capabilities [2].

Machine learning algorithms are excellent at detecting patterns in massive datasets, which is crucial for spotting security concerns. Traditional IDS leverages a variety of machine learning techniques, including k-nearest neighbor, Support Vector Machines (SVM), decision trees [2], [3].

Malware developers' techniques for preventing detection evolve alongside the industry. In this study, we have used the most recent CIC-MalMem-2022 dataset to identify and classify memory-obfuscated malware. This dataset not only helps to detect the presence of malware, but it also provides information on its family and type [4].

As a result, we conducted two experiments: one for binary classification, which determines if a sample is dangerous or benign, and another for multi-class classification, which identifies a specific malware family. To improve the efficiency of these experiments, we used innovative deep learning techniques. We utilized a VGG-16 Convolutional Neural Network (CNN) to convert the information into an image format, letting us extract complicated characteristics from visual representations of malware behavior [5].

The VGG-16's depth and accuracy in image identification make it an excellent candidate for this application. After feature extraction, we are using LightGBM (LGBM), a highly efficient gradient boosting framework, to perform classification tasks. LGMB's capacity to process massive data sets at a cheap computational cost improves our detection system's accuracy and speed [6].

By combining VGG-16 and LGMB into our approach, we aim to create a more robust dynamic malware detection solution with improved accuracy and efficiency compared to traditional methods.

In Section 2, we present a review of previous approaches from the literature. In Section 3, we describe the proposed methodology. In Section 4, we discuss the experiments and their results. In Section 5, we examine into the discussion of our findings. Finally, in Section 6, we provide the conclusion.

2. Literature Review

This section provides an overview of previous studies on popular techniques for selecting features and using machine learning (ML) and deep learning (DL) technologies in intrusion detection systems.

This Research presents that tools have developed like the Static Analyzer for Vicious Executable (SAVE) and Malware Examiner using Disassembled Code (MEDiC) for general malware detection. Their approach promised greater rates of detection with changed malware. However, focusing on static analysis without considering dynamic behaviors creates an enormous research gap. It has used graphical pictures and entropy graphs to detect and categorize malware variants. However, because their approach is based on visualization, it may not be suitable for other types of malwares. [7]

Similarly, this study presented a network intrusion detection system which includes SVM and RF. This strategy uses RF for feature selection, and the KDD Cup 99 dataset has been used to evaluate its effectiveness proposed a feature selection method based on a multilayer perceptron with ordered redundancy. This strategy, which is commonly used for tasks which includes prediction, classification, and regression, is used to discover, and remove unnecessary components. The approach detects network interference via Support Vector Machines (SVM) and Random Forest. RF is used for feature selection, with a dynamic significance technique. Despite using just a few characteristics, the model obtained 93% accuracy on training data, with SVM recommended for scoring.[8]

The growing proliferation of undocumented dangerous software, notably Zero-Day malware, need improved detection systems to avert substantial harm. Zero-Day malware employs complex evasion techniques to prevent detection, forcing further research into efficient identification methods. Machine learning (ML) has emerged as a promising solution, and sandbox settings such as Cuckoo provide a safe arena for experimentation. The suggested Zero-Day Vigilante (Ze Vigilante) system used several ML classifiers, such as Random Forest (RF), Neural Networks (NN), and Support Vector Machine (SVM), to analyze both static and dynamic malware. RF achieved the highest accuracy, with 98.21% for static and 98.92% for dynamic analysis, demonstrating its efficacy [8].

With an increasing number of network-connected devices, such as mobile phones and IoT devices, the potential of security breaches has increased considerably. These systems are becoming more vulnerable to attacks as the number of device kinds increases and the attack surface expands. To address this, security systems often have two layers: a security system, which offers basic protection, and a network intrusion detection system (IDS) or attack detection system, which detects and stops more complex threats. Relying just on a firewall is insufficient, thus malware detection technologies are required for complete protection.[9].

Recent improvements in e-business, e-healthcare, e-governance, and online transactions have provided numerous benefits while increasing the risk of serious cyberattacks. These attacks are intended to disrupt operations, steal critical data, and compromise national defense systems. Cybersecurity solutions are critical for detecting, analyzing, and defending against these attacks. This study examines a variety of assaults, including denial-of-service, botnet, malware, phishing, spoofing, and probing attacks. It focuses on how Machine Learning and Deep Learning approaches tackle these difficulties. Key topics covered include research problems, intrusion detection systems, and the relevance of public and private datasets in cybersecurity research[10].

The Aim of this study is to improve email security by using machine learning techniques to identify spam. Ten distinct machine learning models, including Support Vector Machines, k-Nearest Neighbor, Naïve Bayes, Neural Networks, Recurrent Neural Networks, and others, are used to classify spam emails, which are unsolicited messages. Email data is transformed into a CSV file as part of the process, and this file is subsequently used to train algorithms that identify messages as either spam or "ham" (benign). When evaluated on popular datasets, the method delivers competitive accuracy. Furthermore, the system produces outputs that can be used to enhance spam filtering processes, such as CSV files containing spam IP addresses, their geolocations, and country-specific statistics [11].

This research uses a dataset of malware and good ware samples from Malware Bazaar to propose a dynamic malware analysis and classification method. A dataset was created, features are extracted and scored, six machine learning models are assessed, malware families are classified using Virus Total APIs, and twenty-three distinct types of malware APIs are categorized as part of the five-step process. The Random Forest model yielded the highest results, with high F1-score, AUC, precision, and accuracy. The most serious malware was determined to be ransomware and trojans, and important Windows APIs and system operations for malware detection were noted. In addition to adding additional metrics like AUC and specificity, the strategy raised F1-scores [12]

Traditional static analysis is challenged by malware developers who are always changing their techniques to avoid detection. Dynamic analysis and machine learning together have shown promising results, especially when it comes to detecting Zero-Day malware. The CNN-LSTM algorithm employed in this study has demonstrated potential in mitigating changing cybersecurity risks. With a high accuracy of 96% in identifying malicious activity, the built system—which consists of a log parser, API monitoring, and extension checker—highlights the importance of behavioral analysis and deep learning in cybersecurity[13].

Programs that required conventional identification techniques to complex threats operating at the kernel level, which are more difficult to detect, malware detection has advanced. Traditional techniques utilized CNNs for feature classification or plain text feature extraction alongside machine learning for classification. Modern malware challenges these techniques by frequently displaying familial traits and kernel-level execution. Deep learning is used in many modern solutions. For example, Kim et al. used multi-modal deep learning for Android malware, Droid Detector integrated static and dynamic analysis, and Huang et al. utilized CNNs and sandbox analysis to visualize malware.[14]

With the continuous growth of large data and computational power, deep learning techniques are becoming increasingly common in various fields. In this situation, the researcher suggests employing models based on Recurrent Neural Networks (RNN) for scoring, without the need for pre-training. The performance evaluation was conducted by utilizing the NSL-KDD dataset [Hassan] and the SAP ART training and test set. The evaluation involves comparing various machine learning approaches, such as J48, Support Vector Machine, ANN, Random Forest, and other methods recommended by previous researchers, for the detection of network interference in both binary and multi-class scenarios. Table 1 illustrates the overall literature study of previous techniques.

Table 1. Literature Study

Reference	Survey Outline	Domain	Deep Learning Techniques		
-	-	-	RBM	RNN	CNN
Huang et al. [15]	Software visualization combined with CNNs for dynamic malware analysis.	Malware Detection	No	No	Yes
Linh V, Hùng N et al. [16]	Shallow learning, deep learning, and machine learning and deep				

	learning approaches employed in IDS and related studies have parallels and Differences.	Cyber Security	Yes	No	Yes
Patil R, Deng W et al. [17]	Despite prior efforts and experiences, shallow machine learning approaches have prevented IDS auto encoder (AE) Deployment.	Intrusion detection System	Yes	No	No
Moutafis I, Andreatos A et al. [11]	Deep learning methodologies in IDS are explained in twelve ways, including feature extraction and classification for deletion and classification.	Intrusion detection System	Yes	No	Yes
Chowdhury et al,[18]	Performance and assessment of Machine learning techniques for IDS classification are examined.	Intrusion detection System	Yes	Yes	No
M M, Venkatesh, R. V et al.[10]	Development of efficient threat detection and monitoring systems.	Intrusion detection System	Yes	Yes	No
Hemalatha et al.[19]	Malware classification using DenseNet and data visualization with a reweighted class balance loss function.	Malware Classification	Yes	No	Yes
Sihag V, Vardhan M [7]	Correlation of static and dynamic features using deep learning for Android malware analysis.	Malware Detection and Classification	No	No	Yes

3. Data Set

In this research, leveraging an up-to-date dataset is critical. It is useful for fairly evaluating new methods and determining how well they function in real-world settings. In this experiment, we used the CIC-MalMem-2022 dataset. This collection contains examples of both obfuscated and non-obfuscated malware. To make the study more realistic, it contains popular malware kinds such as spyware, ransomware, and trojans.

We ran two experiments, which include binary classification, which distinguishes between benign and malware samples, and multiclass classification, which detects specific malware kinds. Each sample in the dataset is a memory dump-generated vector of numbers. The key features include Malfind, Ldrmodule, Handles, Procedure View, and Apihooks, for a total of fifty-five features. The dataset contains 58,596 memory dump samples. We divided the data into two sets: training and testing, with training comprising 80% and testing for 20%. Table. 2 illustrates the distribution of Benign and Malware classes and division of training and Testing Dataset.

Table 2. Dataset Distribution for Classification

Class	Train	Test	Total
Benign	23438	5860	29298
Spyware	8016	2004	10020
Ransomware	7833	1958	9791
Trojan	7589	1898	9487

3.1. Converting Malware Binary into Gray Scale Images

To convert the binary files into gray scale images we make use of the hexadecimal representation of the file's binary content and convert those files into PNG images. For example, the resulting image after converting the 0ACDbR5M3ZhBJajygTuf.bytes binary file into a PNG. Figure 1 outlines the steps how we will convert malware binary into gray scale images.

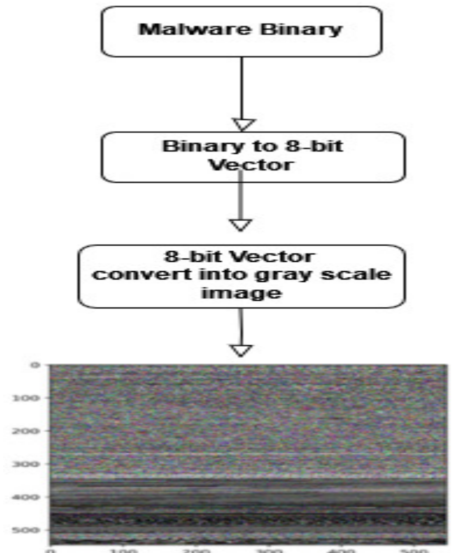

Figure 1. Flowchart to represent conversion of csv dataset into gray scale image**.**

3.2. Images Dataset

In our study, we work with a specially designed image dataset for Malware Detection. The dataset has two directories, one each for training our model and validation sets to validate its performance. Each of the sets comprises 31 classes. As shown in Fig 3.2 an image which belongs to Benign class which has a simple gray scale image with no other noticeable changes in it.

Training set: This directory has been used to train the machine learning model. It contains images labeled with the correct class, letting the model learn and make estimates based on these examples.

Validation Set: Once our model has trained, we need to see how well it performs. This is where the validation set comes in. It contains its own set of labeled images, but different from the images in the training set. Using this separate set, we can ensure that we evaluate the model's ability to detect malware on new, unseen data, which is critical to evaluating its effectiveness.

A closer look at the dataset as presented in Figure 2, our dataset contains images from various categories, including the “Benign” class. An example of an image from this class would be a simple grayscale image that does not have any crucial features or patterns. This simplicity is typical of benign images, which typically do not show the complex characteristics of malware images.

The two categories used to categorize the dataset are benign and malware. Malware frequently changes file’s usual binary structure by using methods like encryption and obfuscation, which purposefully makes code harder to understand. These techniques result in a sudden and unpredictable change in the file's byte sequence.

These byte sequences have visible characteristics like distinct distortions, complicated structures, and sharp lines when converted into images. Because they provide key details regarding the coding methods and malicious behavior, these visual irregularities are essential for the identification and analysis of malware. Researchers and detection systems can examine the structure and characteristics of malware more efficiently according to this visual representation. As Figure 3 illustrates an image belongs to malware class in which pattern we can see distortion which will help us to identify malicious class images.

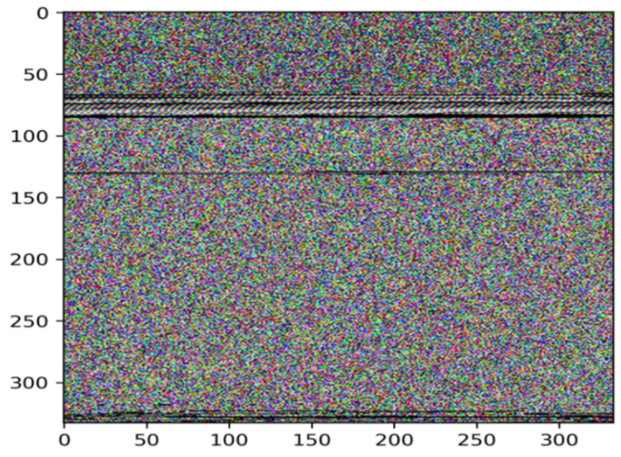

Figure 2. An Image belongs to Benign Class

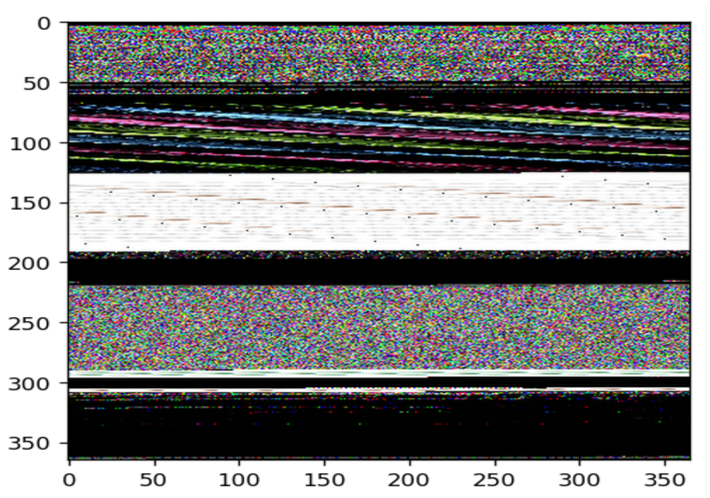

Figure 3. An Image belongs to Malware Class

4. Proposed Methodology for Malware Detection in Wireless Networks

In this study, we create a malware detection system by integrating VGG-16 with the Light Gradient Boosting Machine (LGBM) for feature extraction from an image-based dataset. To begin, the dataset includes labeled images of both benign and malicious software samples obtained from trusted repositories to ensure diversity and completeness. Data preprocessing involves converting malware binaries into standardized grayscale images and scaling them to meet VGG-16 input specifications, ensuring consistency across all samples. Normalizing pixel values as part of feature transformation assures consistency and improves data quality.

We have extracted rich, high-level features from processed images using the VGG-16 convolutional neural network, which has been pre-trained on huge image datasets. These features detect detailed patterns and features that distinguish malicious from benign software. The retrieved feature vectors are then fed into an LGBM model, which was chosen for its efficiency and superior performance in classification tasks. Model selection entails comparing several LGBM configurations to determine the best setting. The labeled dataset has been used to train the model, and the LGBM learns to differentiate between benign and malicious features using iterative optimization. The model's efficacy is determined using metrics such as accuracy, precision, recall, and the F1-score. Figure 4 presents the Architecture of our CNN model will detect and classify benign and malware class images.

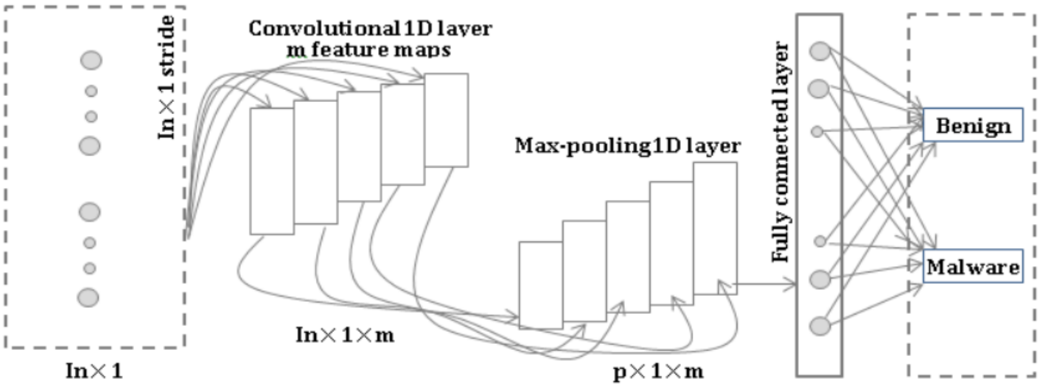

Figure 4. Proposed Architecture of CNN for Malware Detection [20]

Several machine learning models have been built and evaluated on the dataset using several algorithms. Common classifiers such as decision trees, random forests, have been implemented, with a particular emphasis on the recurrent neural network (RNN) model for deep learning. The RNN is designed to process sequential data, most likely memory dumps connected with malware, and make predictions based on the results of the previous time step. Model performance has been assessed using common evaluation parameters such as accuracy, precision, recall, and F1-score. Figure 5 demonstrates the process of Malware Detection.

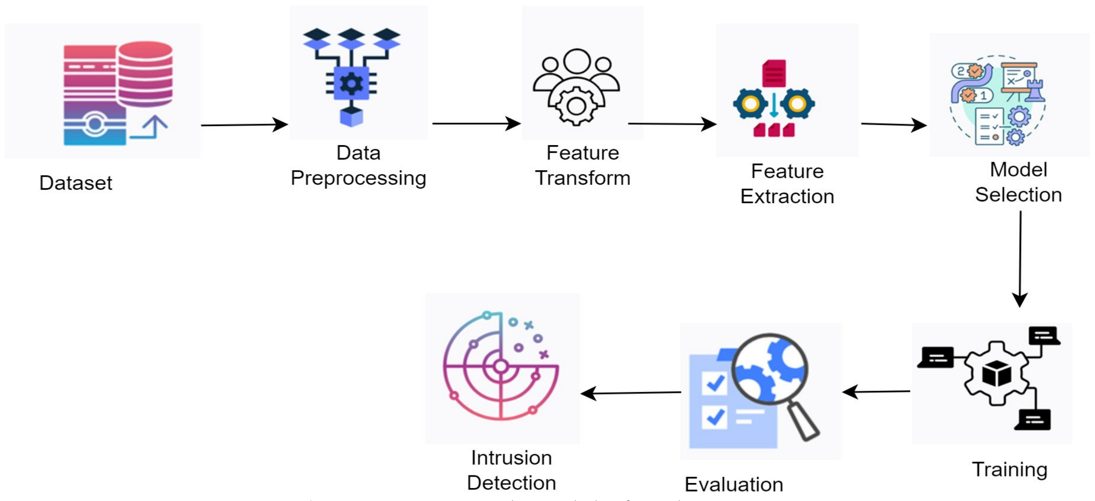

Figure 5. Proposed Model of Malware Detection

4.1. VGG16

VGG-16 is a deep convolutional neural network (CNN) designed for image classification, developed by the Visual Geometry Group at the University of Oxford. VGG16 is a type of CNN (Convolutional Neural Network), which is one of the best computer vision models today. The creators of this model evaluated the networks and increased the depth using an architecture with very small (3 × 3) convolutional filters, which showed a significant improvement over prior art configuration. VGG16 is an object detection and classification algorithm that can classify 1000 images from 1000 different categories with 92.7% accuracy. It is one of the popular image classification algorithms and is easy to use in transfer learning.

4.1.1 Key Features of VGG 16

A VGG network is defined by its depth, which is comprised of 16 layers (VGG16), including convolutional, pooling, and fully connected layers. To maintain spatial resolution, convolutional layers use modest 3x3 filters with a step of one and padding, whereas maximum pooling with 2x2 filters decreases map feature samples. Following the convolution and pooling layers, the network is flattened and connected to fully connected layers, culminating in a softmax layer for classification. Its regular architecture, which consists of repeated convolutional and pooling layers, improves efficiency and ease of usage.

4.1.2. VGG16 Architecture

The architecture has 16 layers, including 13 convolutional layers and three fully connected layers. It has a simple design with blocks of convolutional layers followed by max-pooling layers for down sampling. The network begins with two convolutional layers each with 64 filters, followed by maximum pooling, and then slowly increases the number of filters to 128, 256, and 512 in subsequent layers. Following feature extraction, the output is flattened, and three fully connected layers are used, giving 1000 output classes. An Architecture of VGG 16 for Features Extraction in which we give a malware class image (224*224*3) as an input.

4.2. Pretrained CNN Models for features extraction and images classification

We have used a convolutional neural network (CNN) VGG16, to extract features from images. These features are typically the outputs of one of the final layers before the classification layer. Once you have these features, we have used them as input to the LGBM Classifier. By combining LGBM with a feature extraction method VGG16, we have leveraged its strengths in handling structured data while working with image datasets. LightGBM may be efficiently combined with convolutional neural networks (CNNs), such as VGG16, for image classification. VGG16 is well-known for its ability to extract detailed representations of features from images using deep architecture and convolutional layers. High-level features are retrieved from the images using VGG16 before being fed into the LightGBM model. LightGBM's efficient tree algorithms analyze these feature vectors for classification tasks. This hybrid technique enhances classification accuracy while maximizing LightGBM's efficiency in large-scale data processing.

5. Implementation and Result Analysis

We achieved significant results by successfully implementing various machine learning techniques, such as Support Vector Machine (SVM), Random Forest, and LGBM Classifier, for both binary and multiclass classification tasks with the CSV dataset, as well as using the VGG-16 model for feature extraction from our image dataset. For the picture dataset, we used the LGBM Classifier to perform classification, allowing us to accurately group the photos into distinct classes.

The classification process for the CSV dataset was thorough, going beyond the usual benign and malicious categories. The classifiers not only accurately distinguished between benign and malware samples, but they also classified the malware into specific sub-classes. This vast classification provides a better understanding of the many forms of malware present and improves the system's capacity to detect and respond to different malware threats more precisely. Overall, the combination of these methodologies has proved the ability to manage and analyze large datasets, resulting in more precise and accurate malware identification and categorization.

5.1. Number of Occurrences of each Malware Class

There are categorized the malware into main classes 1) Benign has occurrence 29298, 2) Spyware has occurrence 10020, 3) Ransomware has occurrence 9791 and 4) Trojan has occurrence 9487 as shown in Figure 6.

Principal Component Analysis (PCA) is a dimension reduction approach for projecting high-dimensional data into a lower-dimensional space while optimizing for variance conservation. In a scatter plot, the x- and y-axes represent the first two principal components, which represent the data set's directions of highest variance. The data points are color-coded, with blue dots indicating benign samples and red dots signifying malware samples. The overlap and clustering of both benign and malware samples in the plot demonstrate that the first two main components do not successfully discriminate between the two classes in this dataset.

The graph demonstrates the use of **t-Distributed Stochastic Neighbor Embedding (t-SNE)**, a nonlinear dimensionality reduction method, to show high-dimensional data in two dimensions. The x and y axes in this t-SNE plot do not represent features, but rather modified data coordinates for viewing reasons. Data points are color-coded, with blue dots indicating benign samples and red dots indicating malware samples. The plot shows greater levels of clustering between benign and malware samples than the principal component analysis (PCA) plot, indicating that t-SNE successfully grabbed some of the fundamental patterns in the data that distinguish these sample types. Although this improvement, some intersection throughout clusters remains.

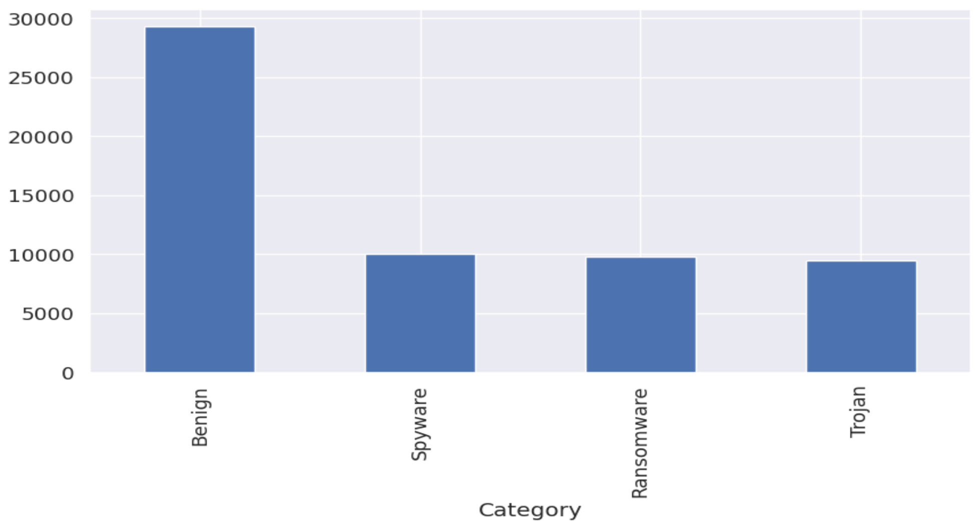

Figure 6. Number of occurrences of each malware class

Linear Discriminant Analysis (LDA) is a supervised reducing the dimensional method for classification tasks. It creates a linear combination of features that maximizes the distinction of various classes. The LDA in Figure depicts this separation along its axes, which reflect the linear discriminants the directions that maximize between-class variance while reducing within-class variability.

The graph shows blue and red points representing benign and malware samples, respectively. The graph clearly differentiates between the two classes, showing that LDA has selected the projection that best separates benign and malware samples in the dataset. This effective separation highlights the value of LDA for improving class discrimination. Figure 7 reveals the visualization of class distribution of Benign and Malware.

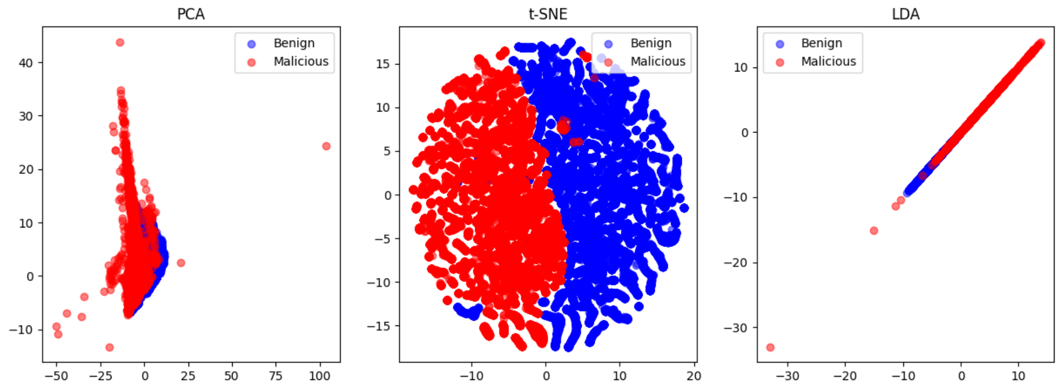

Figure 7. Visualization of Class Separation Using PCA, t-SNE, and LDA for Malware Detection

It not only classifies the number of occurrences of each malware class, but it also categorizes Malware into subclasses and tells us whether the malware is Trojan, spyware, or ransomware.

It not only monitors the prevalence of each malware class, but it also goes deeper, categorizing the malware into more specific subclasses. Furthermore, it provides precise information about the type of malware, indicating whether it is *Trojan, spyware, or ransomware*, and identifying various kinds within these categories. This comprehensive approach improves comprehension of the malware's nature and functionality, allowing for more targeted and effective responses to varied threats.

After implementing Machine Learning Classifiers and LGBM Classifier, we obtained the anticipated results with a high accuracy of 0.9997 and a high recall rate. After implementing our classifiers, we obtained high values for Balanced Accuracy and Mathews Correlation Coefficient as shown in Figure 8.

The left graph shows the loss values for training and testing datasets over ten epochs. The training loss, represented by the blue line, decreases steadily across epochs, indicating good model training and increasing error reduction on the training dataset. In contrast, the test loss, represented by the orange line, largely follows the trend of the training loss, but increases significantly in epoch 6. This spike may indicate over fitting or the effect of data noise during this period, which is followed by a significant reduction in loss.

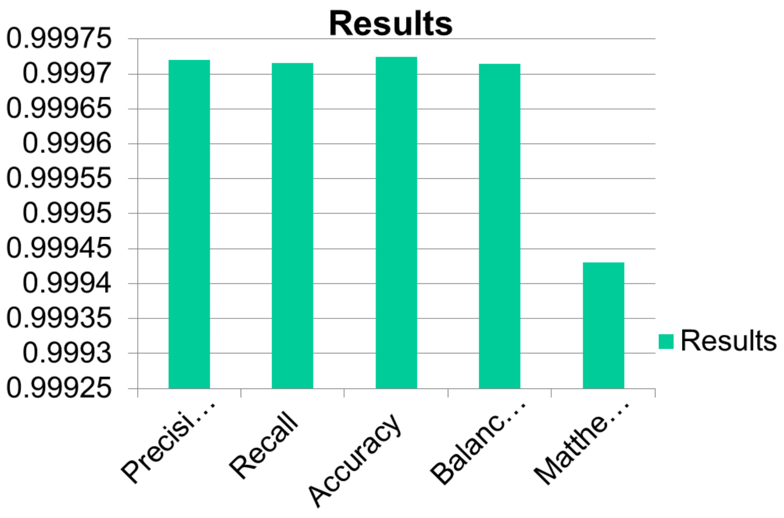

Figure 8. Model's Training and Testing Accuracy, Loss and AUC scores

. **The Middle graph** shows the model's accuracy on both the training and test datasets within epochs. The blue line indicates training accuracy, which consistently improves during the training process, telling improved performance on the training data. The testing accuracy, represented by the orange line, increases overall but decreases at epoch 6, which connects to the testing loss peak. Following this fall, testing accuracy quickly improves, exceeding training accuracy. This random pattern could indicate a variation problem or that the test set is easier than the training set.

The graph illustrates the **area under the curve (AUC)** scores for both the training and test datasets across multiple epochs. The training AUC, shown in the blue line, keeps rising, showing that the model's ability to discriminate between classes is gradually improving. In contrast, the tested AUC, displayed by the orange line, shows a rising trend, although with a fluctuating fall in epoch 6 before a strong climb. This tendency corresponds to the fluctuations observed in the accuracy and loss measurements. Figure 9 shows the graph of Model training and Testing Accuracy.

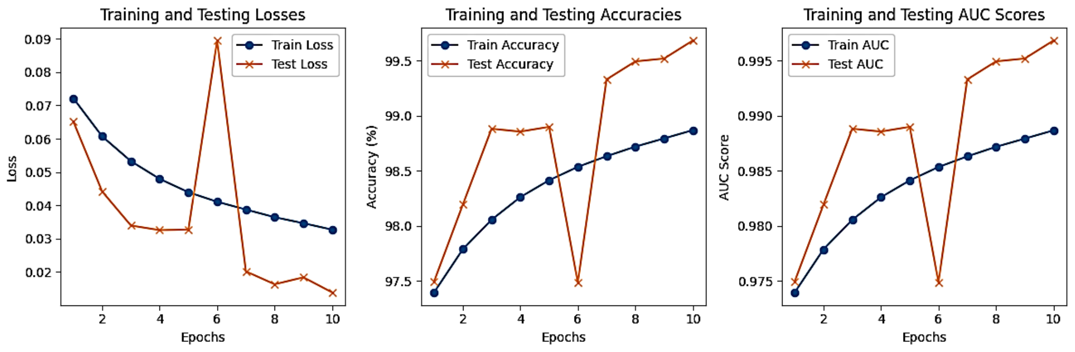

Figure 9. Model's Training and Testing Accuracy graph

5.2. Performance Evaluation Metrics

To further investigate the false positive and false negative cases, we created a confusion matrix analysis to illustrate my method's classification performance on both datasets. The accuracy metric is the main performance indicator used for evaluation in this model. True positive (TP) refers to instances that are accurately classified as positive, whereas false negative (FN) refers to situations that are overlooked as positive, suggesting abnormalities. False positive (FP) refers to cases that are incorrectly detected as positive, whereas true negative (TN) implies instances that are correctly identified as negative. A classification model's accuracy is the ratio of correctly predicted instances (including true positives and true negatives) to the total number of instances. as shown in equation 1.

$$\text{Accuracy}=\frac{TP+TN}{TP+FP+TN+FN} \quad (1)$$

$$\text{Precision}=\frac{TP}{TP+FP} \quad (2)$$

$$\text{Recall}=\frac{TP}{TP+FN} \quad (3)$$

$$\text{F1 Score}= \mathbf{2} \times \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

Precision assesses the accuracy of positive predictions by indicating the fraction of true positives among all predicted positives. As shown in equation 2. Recall (also known as sensitivity or true positive rate) measures a model's ability to correctly identify cases that are positive. It is the ratio of true positives to the sum of true positives plus false negatives. As shown in equation 3. F1 Score is a metric that balances precision and recall, providing a single way to evaluate a model's performance while compensating for both false positives and false negatives. It is the harmonic mean of precision and recall. As shown in equation 4.

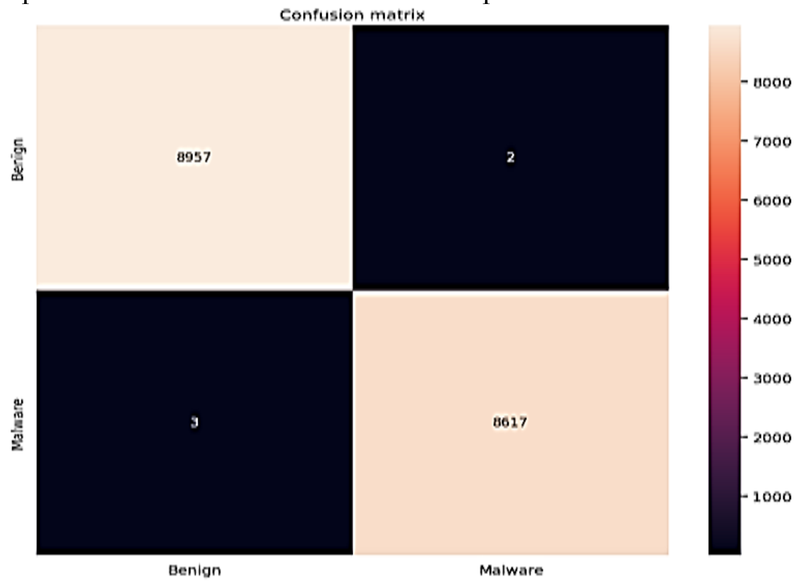

Figure 10. Confusion Matrix for Binary Classification

In Figure 10, a confusion matrix is for a binary classification problem with two classes: **Benign** and **Malware**.

Table 4. Benign and malware distribution in binary classification

Parameter	Predicted: Benign	Predicted: Malware
Actual: Benign	8957	2
Actual: Malware	3	8617

There were 8617 instances of true malware that were correctly identified as malware. Furthermore, 8957 benign occurrences were correctly identified as benign and shown in Table 4.

However, two benign cases were wrongly classified as malware, causing false alarms. Furthermore, three malware cases were wrongly classified as benign, resulting in missed detections. In Figure 11 a confusion matrix represents a multiclass classification problem with four classes: Benign, Ransomware, Spyware, and Trojan.

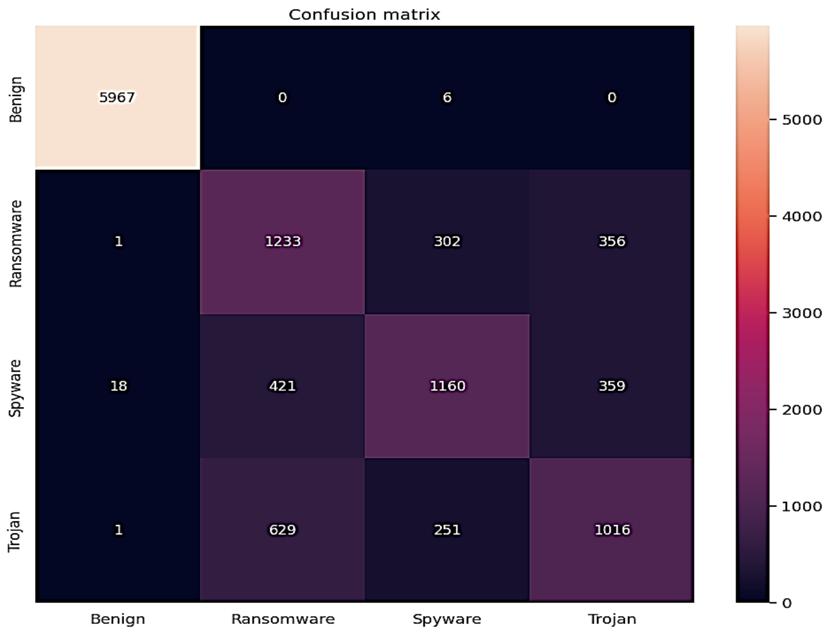

Figure 11. Confusion Matrix for Multiclass Classification

Table 5. A breakdown of the confusion matrix

Parameters	**Predicted: Benign**	**Predicted: Ransomware**	**Predicted: Spyware**	**Predicted: Trojan**
Actual: Benign	5967	0	6	0
Actual: Ransomware	1	1233	302	356
Actual: Spyware	18	421	1160	359
Actual: Trojan	1	629	251	1016

In the Benign class, 5967 occurrences were correctly classified as benign, with just 6 misclassified as spyware. In the Ransomware category, 1233 occurrences were correctly classified, but one instance was wrongly classified as benign, 302 as spyware, and 356 as a Trojan. The Spyware class has 1160 instances successfully detected, though 18 were misclassified as benign, 421 as ransomware, and 359 as Trojans. In the Trojan category, 1016 instances were precisely classified, however there were some inaccuracies, with one instance classified as benign, 629 as ransomware, and 251 as spyware shown in Table 5.

Key findings from the analysis show that the benign class is primarily correctly identified, with a low rate of misclassification. This indicates the model is quite good at identifying benign from malicious cases. Certain varieties of malware, however, face additional hurdles. For example, ransomware can occasionally be confused with spyware or Trojan horses, despite the reality that a large percentage of ransomware cases are correctly recognized.

In the case of spyware, the model has a high percentage of proper identification; however, there is a significant issue with misclassification, with some spyware occurrences being wrongly classified as ransomware or Trojan horse. Similarly, while the Trojan class illustrates a great number of valid classifications, it suffers from a substantial amount of misclassification, a few numbers of Trojans are mistakenly classified as ransomware or spyware. These results present classification model has possibilities for additional modification to increase accuracy across all malware categories.

In Figure 12 confusion matrix represents a multiclass classification scenario with 31 separate classes, with each cell representing how many instances of a specific class were assigned to their predicted classes. The matrix mainly displays correct classifications along the diagonal, indicating effective performance. For example, Class 0 had 74 correct predictions, Class 1 had 43, and Class 2 had 54, indicating high overall performance. However, there were many misclassifications: class 1 instances were misclassified as classes 2, 3, and 30; Class 8 had 8 cases misclassified as class 7; Class 15 had one case misclassified as class 8; and class 29 had one instance misclassified as class 7. These errors indicate that the model may have difficulty separating the different classes.

Some classes, such as class 18 and class 20, received fewer correct classifications, indicating that they may be difficult to correctly identify. In addition, class 24 was misclassified into classes 0 and 29. Overall, the model is quite accurate, with few misclassifications. To calculate precision and other metrics such as precision, recall, and F1 score, add the correct classifications along the diagonal and divide by the total number of instances.

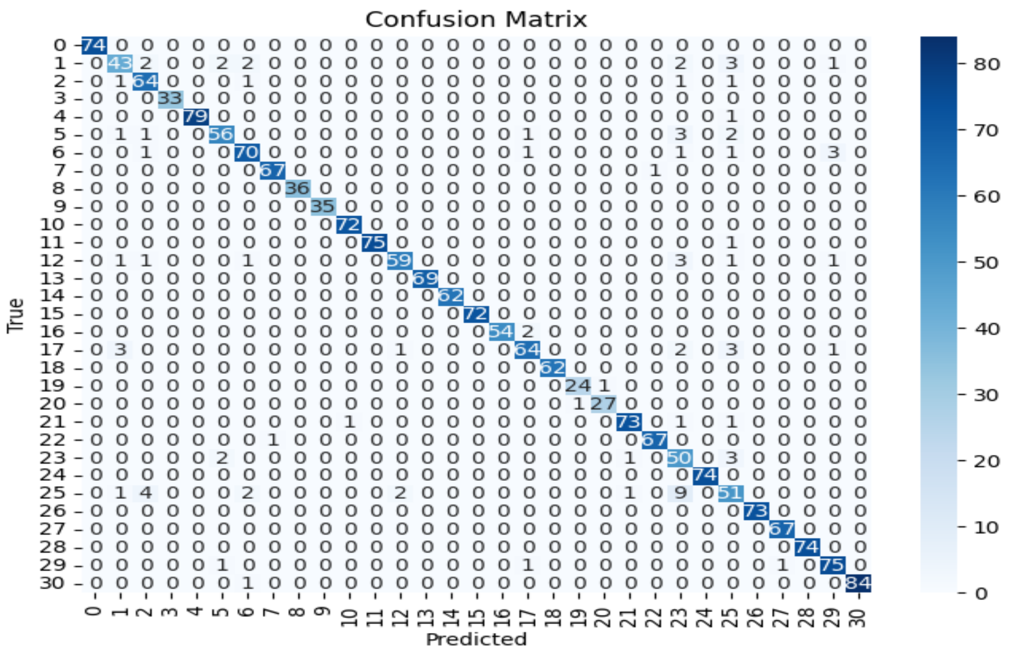

Figure 12. Confusion matrix for the classification of images dataset using the VGG-16 model with LightGBM (LGBM).

In this research, both CSV and images datasets were used to assess the efficiency of several classification approaches for dynamic malware detection. Traditional machine learning models were used to categorize benign and malicious instances in the CSV dataset, and the findings of the confusion matrix show that they performed well. On the other hand, the images dataset required a more complex technique due to the need for feature extraction, which was successfully handled by the VGG-16 model. The collected features were then classified with LightGBM (LGBM), which resulted in excellent precision in identifying distinct malware classifications. While the model performed well generally, several misclassifications were detected, notably amongst visually similar malware groups. These findings imply that, while the combination of deep learning and gradient boosting approaches is effective, more refining could improve accuracy, particularly when discriminating across closely related malware types. This holistic approach emphasizes the importance of combining classic and modern cybersecurity technologies for successful threat identification.

7. Conclusion and Future Work

This research successfully constructed a dynamic malware detection system by combining CSV and images information and employing deep learning techniques to improve malware detection accuracy and robustness. The approach took advantage of the benefits of traditional machine learning models for CSV data, employing the VGG-16 model for feature extraction from images, followed by classification with LightGBM (LGBM). The results illustrate that this hybrid methodology has enormous potential, with near-perfect performance in binary classification tasks and strong results in multiclass classifications. The system's ability to properly identify several types of malwares while minimizing false positives and false negatives demonstrates the value of combining several data sources with advanced feature extraction techniques. This research emphasizes the importance of dynamic analysis and deep learning in improving cybersecurity processes, providing a more dependable and efficient solution for detecting and facing sophisticated and developing malware attacks. Future research in dynamic malware detection using deep learning algorithms will concentrate on practical improvements. Integrating other data sources, such as network activity and user behavior, may provide a deeper understanding of malware operations, resulting in increased detection capabilities. Furthermore, efforts will be made to increase the interpretability and transparency of these models, allowing them to be effectively used in real-life situations where trust and understanding are critical. These developments will help to create more flexible, resilient, and efficient malware detection systems, more suitable to the difficulties of an ever-changing cybersecurity landscape.

Acknowledgments: We want to thank everyone who helped us with this research. And a big high-five to University of Engineering and Technology, Taxila for letting us uses their facilities and stuff. And a special shout out to all the people who took part in our study. We really appreciate your time and effort. And a big thanks to the reviewers who gave us helpful feedback to make our paper better. Lastly a huge thanks to our family and friends for always cheering us on.

References

1. B. Godbin and S. G. Jasmine, "Enhancing multiclass pneumonia classification with Machine Learning and textural features," Machine Graphics and Vision, vol. 32, no. 3/4, pp. 83–106, Dec. 2023, doi: 10.22630/MGV.2023.32.3.5.
2. Brown, M. Gupta, and M. Abdelsalam, "Automated machine learning for deep learning based malware detection," Comput Secur, vol. 137, p. 103582, Feb. 2024, doi: 10.1016/j.cose.2023.103582.
3. L. Shen et al., "Self-attention based convolutional-LSTM for android malware detection using network traffics grayscale image," Applied Intelligence, vol. 53, no. 1, pp. 683–705, Jan. 2023, doi: 10.1007/s10489-022-03523-2.
4. Mezina and R. Burget, "Obfuscated malware detection using dilated convolutional network," in International Congress on Ultra Modern Telecommunications and Control Systems and Workshops, IEEE Computer Society, 2022, pp. 110–115. doi: 10.1109/ICUMT57764.2022.9943443.
5. W. Al-Khater and S. Al-Madeed, "Using 3D-VGG-16 and 3D-Resnet-18 deep learning models and FABEMD techniques in the detection of malware," Alexandria Engineering Journal, vol. 89, pp. 39–52, Feb. 2024, doi: 10.1016/j.aej.2023.12.061.
6. K. Kosmidis and C. Kalloniatis, "Machine Learning and Images for Malware Detection and Classification," in Proceedings of the 21st Pan-Hellenic Conference on Informatics, New York, NY, USA: ACM, Sep. 2017, pp. 1–6. doi: 10.1145/3139367.3139400.
7. V. Sihag, M. Vardhan, P. Singh, G. Choudhary, and S. Son, "De-lady: Deep learning based android malware detection using dynamic features," Journal of Internet Services and Information Security, vol. 11, no. 2, pp. 34–45, May 2021, doi: 10.22667/JISIS.2021.05.31.034.
8. F. Alhaidari et al., "ZeVigilante: Detecting Zero-Day Malware Using Machine Learning and Sandboxing Analysis Techniques," Comput Intell Neurosci, vol. 2022, pp. 1–15, May 2022, doi: 10.1155/2022/1615528.
9. T. A. Jasi and M. M. T. Jawhar, "Detecting network attacks Model based on a long short-term memory LSTM," Technium: Romanian Journal of Applied Sciences and Technology, vol. 4, no. 8, pp. 64–72, Aug. 2022, doi: 10.47577/technium.v4i8.7225.
10. M. M, Venkatesh, and V. K. R., "Cyber Security Threats and Countermeasures using Machine and Deep Learning Approaches: A Survey," Journal of Computer Science, vol. 19, no. 1, pp. 20–56, Jan. 2023, doi: 10.3844/jcssp.2023.20.56.
11. Moutafis, A. Andreatos, and P. Stefaneas, "Spam Email Detection Using Machine Learning Techniques," European Conference on Cyber Warfare and Security, vol. 22, no. 1, pp. 303–310, Jun. 2023, doi: 10.34190/eccws.22.1.1208.
12. D. Z. Syeda and M. N. Asghar, "Dynamic Malware Classification and API Categorisation of Windows Portable Executable Files Using Machine Learning," Applied Sciences, vol. 14, no. 3, p. 1015, Jan. 2024, doi: 10.3390/app14031015.
13. G. Karat, J. M. Kannimoola, N. Nair, A. Vazhayil, S. V G, and P. Poornachandran, "CNN-LSTM Hybrid Model for Enhanced Malware Analysis and Detection," Procedia Comput Sci, vol. 233, pp. 492–503, 2024, doi: 10.1016/j.procs.2024.03.239.
14. G. Karat, J. M. Kannimoola, N. Nair, A. Vazhayil, S. V G, and P. Poornachandran, "CNN-LSTM Hybrid Model for Enhanced Malware Analysis and Detection," Procedia Comput Sci, vol. 233, pp. 492–503, 2024, doi: 10.1016/j.procs.2024.03.239.
15. Shah, A. M., Aljubayri, M., Khan, M. F., Alqahtani, J., Sulaiman, A., & Shaikh, A. (2023). ILSM: Incorporated Lightweight Security Model for Improving QOS in WSN. Computer Systems Science & Engineering, 46(2).
16. Y. Liu, H. Fan, J. Zhao, J. Zhang, and X. Yin, "Efficient and Generalized Image-Based CNN Algorithm for Multi-Class Malware Detection," IEEE Access, 2024, doi: 10.1109/ACCESS.2024.3435362.
17. V. K. Linh, N. V. Hùng, T. N. Anh, D. Do Nhuan, and D. C. Hien, "ENHANCE DEEPLEARNING MODEL FOR MALWARE DETECTION WITH A NEW IMAGE REPRESENTATION METHOD," Journal of Science and Technology on Information security, pp. 31–39, Jun. 2024, doi: 10.54654/isj.v1i21.1000.
18. R. Patil and W. Deng, "Malware analysis using machine learning and deep learning techniques," in Conference Proceedings - IEEE SOUTHEASTCON, Institute of Electrical and Electronics Engineers Inc., Mar. 2020. doi: 10.1109/SoutheastCon44009.2020.9368268.
19. Md. A. Talukder et al., "A Dependable Hybrid Machine Learning Model for Network Intrusion Detection," Dec. 2022, doi: 10.1016/j.jisa.2022.103405.
20. P. Thakur, V. Kansal, and V. Rishiwal, "Hybrid Deep Learning Approach Based on LSTM and CNN for Malware Detection," Wirel Pers Commun, vol. 136, no. 3, pp. 1879–1901, Jun. 2024, doi: 10.1007/s11277-024-11366-y.

21. R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust Intelligent Malware Detection Using Deep Learning," IEEE Access, vol. 7, pp. 46717–46738, 2019, doi: 10.1109/ACCESS.2019.2906934.
22. S. Kumar and A. Kumar, "Image-based malware detection based on convolution neural network with autoencoder in Industrial Internet of Things using Software Defined Networking Honeypot," Eng Appl Artif Intell, vol. 133, p. 108374, Jul. 2024, doi: 10.1016/j.engappai.2024.108374.
23. Murtza, I., Saadia, A., Basri, R., Imran, A., Almuhaimeed, A., & Alzahrani, A. (2022). Forex investment optimization using instantaneous stochastic gradient ascent—Formulation of an adaptive machine learning approach. Sustainability, 14(22), 15328. MDPI.
24. Ashfaq, A., Imran, A., Ullah, I., Alzahrani, A., Alheeti, K. M. A., & Yasin, A. (2022). Multi-model ensemble based approach for heart disease diagnosis. In 2022 International Conference on Recent Advances in Electrical Engineering & Computer Sciences (RAEE & CS) (pp. 1-8). IEEE.