Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.063448

ARTICLE

Advanced Techniques for Dynamic Malware Detection and Classification in Digital Security Using Deep Learning

Taher Alzahrani*

Information System Department, College of Computer and Information Science, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, 11673, Saudi Arabia

*Corresponding Author: Taher Alzahrani. Email: talzahrani@imamu.edu.sa

Received: 15 January 2025; Accepted: 25 March 2025; Published: 19 May 2025

ABSTRACT: The rapid evolution of malware presents a critical cybersecurity challenge, rendering traditional signature-based detection methods ineffective against novel variants. This growing threat affects individuals, organizations, and governments, highlighting the urgent need for robust malware detection mechanisms. Conventional machine learning-based approaches rely on static and dynamic malware analysis and often struggle to detect previously unseen threats due to their dependency on predefined signatures. Although machine learning algorithms (MLAs) offer promising detection capabilities, their reliance on extensive feature engineering limits real-time applicability. Deep learning techniques mitigate this issue by automating feature extraction but may introduce computational overhead, affecting deployment efficiency. This research evaluates classical MLAs and deep learning models to enhance malware detection performance across diverse datasets. The proposed approach integrates a novel text and image-based detection framework, employing an optimized Support Vector Machine (SVM) for textual data analysis and EfficientNet-B0 for image-based malware classification. Experimental analysis, conducted across multiple train-test splits over varying timescales, demonstrates 99.97% accuracy on textual datasets using SVM and 96.7% accuracy on image-based datasets with EfficientNet-B0, significantly improving zero-day malware detection. Furthermore, a comparative analysis with existing competitive techniques, such as Random Forest, XGBoost, and CNN-based (Convolutional Neural Network) classifiers, highlights the superior performance of the proposed model in terms of accuracy, efficiency, and robustness.

KEYWORDS: Machine learning; EffiicientNet B0; malimg dataset; XceptionNet; malware detection; deep learning techniques; support vector machines (SVM)

1 Introduction

The modern information society has developed as a result of the Internet of Things (IoT) and its associated applications. However, achieving the full benefits of this industrial revolution is severely hampered by security concerns. Cybercriminals frequently attack individual computers and networks in order to denial-of-service systems and steal private information for financial gain. Malicious software, or malware, is what these attackers utilize to threaten and weaken systems seriously. A computer software called malware is made to damage an operating system (OS). Malware runs by many names, such as adware, spyware, virus, worm, trojan, rootkit, backdoor, ransomware, and command and controls C2 bot, depending on its behavior and goal.

Malware detection and mitigation are still developing issues in cybersecurity. Malware writers always enhance their strategies to avoid detection as researchers create new ones. Our goal in this investigation was to identify and categorize memory-based obfuscated malware. For my experiments, I utilized the most recent dataset, the Canadian Institute for Cybersecurity Malware Memory Analysis 2022 (CIC-MalMem-2022). This dataset offers information about the kind and family of malware in addition to detecting its existence. Therefore, we ran two experiments: one to identify the malware family (multi-class classification) and the other to assess whether a particular sample is harmful (binary classification). To do this, we have evaluated several conventional machine learning techniques, such as logistic regression, decision trees, random forests, multi-layer perceptron (MLP), and support vector machines (SVM). We applied random search as an optimization technique to find the best hyperparameters for these methods. 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 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 a k-nearest neighbor, Support Vector Machines (SVM), and decision trees [3,4]. 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 also provides information on its family and type [5,6]. 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. After feature extraction, we are using LightGBM (LGBM), a highly efficient gradient boosting framework, to perform classification tasks. LGBM capacity to process massive data sets at a cheap computational cost improves our detection system's accuracy and speed [7].

We aim to enhance the maturity of cybersecurity through the development of advanced deep-learning models. Our approach involves both Comma-Separated Values (CSV) and image datasets, utilizing the Visual Geometry Group 16-layer (VGG-16) model for image analysis. By focusing on dynamic malware behaviours, our objective is to achieve real-time detection capabilities that outperform traditional methods. This study not only addresses current cyber security challenges but also contributes novel insights into utilizing machine learning for proactive threat mitigation. Our findings aim to pave the way for more effective, scalable, and adaptive malware detection systems. The rapid proliferation of malware, with thousands of new variants appearing each day, poses a significant threat to cyber security. Traditional static analysis methods struggle to keep pace with these evolving threats. The contribution of the study is that various deep learning and machine learning approaches, which are frequently used in many intrusion detection systems in the literature and are very popular, have been confirmed to achieve successful results in malware detection. The major contributions of this research are as follows:

- The framework proposed for dynamic malware detection framework that addressed the challenges in this field by analyzing the behavior of malware in real time, allowing for the classification of samples into families based on their operational characteristics. The proposed system for dynamic malware detection classifies samples into families according to their operational features by analyzing malware's static and dynamic behavior in real-time.
- We carried out extensive experimentation on picture texture analysis using modern deep learning algorithms. These trials show considerable increases in model performance over typical feature extraction techniques.

- We rigorously validate both widely used machine leaning and less explored deep learning models in the malware classification domain, utilizing a comprehensive benchmark malware dataset. This approach ensures a robust comparison of our proposed solution against state-of-the-art methods, providing a fair and accurate assessment of its performance.
- This method makes it possible to create focused mitigation techniques that may successfully eradicate treats. This significantly bolsters cybersecurity defenses, offering a scalable and adaptive solution to combat evolving threats.

The author's contributions facilitated the advantages of this research in analyzing malware behavior in real-time enabling immediate identification and mitigation of emerging threats significantly, enhancing cybersecurity responsiveness and reducing the risk of infection. Utilizing advanced deep learning techniques and extensive experiments on picture texture results in higher accuracy and reliability. Real-time malware behavior monitoring allows immediate detection and protection of new threats, producing stronger cybersecurity responses and decreasing the chances of attack success. The combination of deep learning technology with extensive picture texture testing produces more precise malware detection that reduces wrong positive and negative results. The use of benchmark datasets during validation enables researchers to compare their methods against established standards while building solid foundations for future investigations.

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 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 DL technologies in intrusion detection systems. This research presents that tools have been developed, such as the Static Analyzer for Vicious Executables (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 malware [8].

Similarly, this study presented a network intrusion detection system that includes SVM and RF. This strategy uses RF for feature selection, and the KDD Cup 99 dataset has been used to evaluate its effectiveness and proposed a feature selection method based on a multi-layer perceptron with ordered redundancy. This strategy, which is commonly used for tasks that include 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 [9].

The growing proliferation of undocumented dangerous software, notably Zero-Day malware, needs 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 [9].

With an increasing number of network-connected devices, such as mobile phones and IoT devices, the potential for 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. The 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 only on a firewall is insufficient; thus, malware detection technologies are required for complete protection [10]. Data balancing, a novel generative adversarial network model, has been put out [11]. Malware detection is modeled using heterogeneous graphs. A novel technique is presented to mitigate the over-smoothing issue. 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 [12].

This study aims 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 [13].

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 were extracted and scored, six machine learning models were assessed, malware families were classified using Virus Total Application Programming Interface (APIs), and twenty-three distinct types of malware APIs were 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 [14].

Traditional static analysis is challenged by malware developers who are always changing their techniques to avoid detection. Together, dynamic analysis and machine learning have shown promising results, especially when it comes to detecting zero-day malware. The Convolutional Neural Networks and Long Short-Term Memory (CNN-LSTM) algorithm employed in this study has demonstrated the potential to mitigate 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 [15]. Programming 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 (Convolutional Neural Networks) 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 [15].

Deep learning techniques are becoming increasingly common in various fields with the continuous growth of large data and computational power. 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 Systems, Applications, and Products (SAP) in Data Processing, training and test set. The evaluation involves comparing various machine learning approaches, such as J48, Support Vector Machine, Artificial Neural Network (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: Critical literature summary

Reference	Survey outline	Domain	Deep learning techniques		
			Restricted Boltzmann Machine (RBM) RBM	**RNN**	**CNN**
Liu et al. [16]	Software visualization combined with CNNs for dynamic malware analysis.	Malware detection	No	No	Yes
Linh et al. [17]	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 et al. [18]	Despite prior efforts and experiences, shallow machine learning approaches have prevented IDS autoencoder (AE) deployment.	Intrusion detection system	Yes	No	No
Moutafis et al. [13]	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

(Continued)

Table 1 (continued)

Reference	Survey outline	Domain	Deep learning techniques		
			Restricted Boltzmann Machine (RBM) RBM	RNN	CNN
Talukder et al. [19]	Performance and assessment of machine learning techniques for IDS classification are examined.	Intrusion detection system	Yes	Yes	No
Jasi et al. [10]	Development of efficient threat detection and monitoring systems.	Intrusion detection system	Yes	Yes	No
Thakur et al. [20]	Malware classification using DenseNet and data visualization with a reweighted class balance loss function.	Malware classification	Yes	No	Yes DenseNet
Sihag et al. [8]	Correlation of static and dynamic features using deep learning for Android malware analysis.	Malware detection and classification	No	No	Yes

The majority of authors in the literature rely on static and dynamic analysis using pre-established signatures, yet this method misses new malware variants, leaving systems vulnerable. Conventional machine learning algorithms require human engineers to manually develop features, which hinders their real-time performance and reduces their speed in dynamic systems. Deep learning offers improved accuracy and automatic feature extraction, but it also poses serious computational problems that limit its use in real-time applications and situations with limited resources. The literature focuses on deep learning approaches in cybersecurity, including CNNs for dynamic malware analysis. The authors used CNNs for dynamic malware analysis, whereas Sihag and Vardhan used static and dynamic characteristics for Android malware analysis. Some authors investigated shallow and deep learning methods in IDS. The literature addressed limitations in IDS autoencoder deployment. It examines several deep-learning approaches for IDS. In the literature, DenseNet was used for malware classification. This research, combined, highlights the efficiency of deep learning in improving the accuracy and resilience of cybersecurity systems. It also emphasises the potential for future advancements in cybersecurity systems.

3 Data Set Details

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. Its 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. The Malimg dataset includes images converted from malware binaries, typically resized to a standard dimension of 64 × 64 pixels. This standardization ensures consistency and efficiency in processing while retaining essential features for analysis. The size of each image significantly impacts the processing speed and complexity of the proposed scheme. Larger images contain more pixels, resulting in higher computational requirements for processing and analysis. It can slow down the processing speed, especially in real-time applications.

The CIC-MalMem-2022 dataset simulates real-world scenarios with 58,596 records, balanced between 50% malicious and 50% benign memory dumps. It includes various malware families like Spyware, Ransomware, and Trojan Horse. Feature selection is based on memory dump analysis to detect obfuscated malware. Data imbalance is addressed by ensuring an equal distribution of benign and malicious samples.

The Malimg dataset focuses on image-based malware detection. Feature selection involves extracting visual patterns from malware images. Data imbalance issues are mitigated using techniques like data augmentation to balance the dataset. This ensures the model can effectively learn from both minority and majority classes.

We managed two experiments, which include binary classification, which distinguishes between benign and malware samples, and multi-class 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 the division of training and Testing Datasets.

Table 2: Dataset distribution for classification

Sr. No.	Class	Train	Test	Total
1	**Benign**	23,438	5860	29,298
2	**Trojan**	7589	1898	9487
3	**Ransomware**	7833	1958	9791
4	**Spyware**	8016	2004	10,020

3.1 Converting Malware Binary into Gray Scale Images

The binary files are converted into PNG pictures using the hexadecimal representation of the binary content to create grayscale graphics. For instance, the final image is produced when a binary file is converted to a PNG. Fig. 1 outlines the steps of how we convert malware binary into grayscale images using an 8-bit vector.

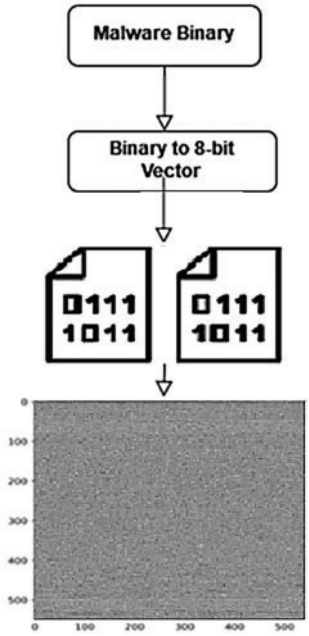

Figure 1: Flowchart to represent the conversion of CSV dataset into a grayscale image

3.2 Images Dataset

In our study, we work with a specially designed image dataset for Malimg Malware Detection. The dataset has two directories for training our model and validation sets to validate its performance. Each of the sets comprises 25 classes [21]. As shown in Figs. 2 and 3, an image that belongs to the Benign class has a simple grayscale image with no other noticeable changes.

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.

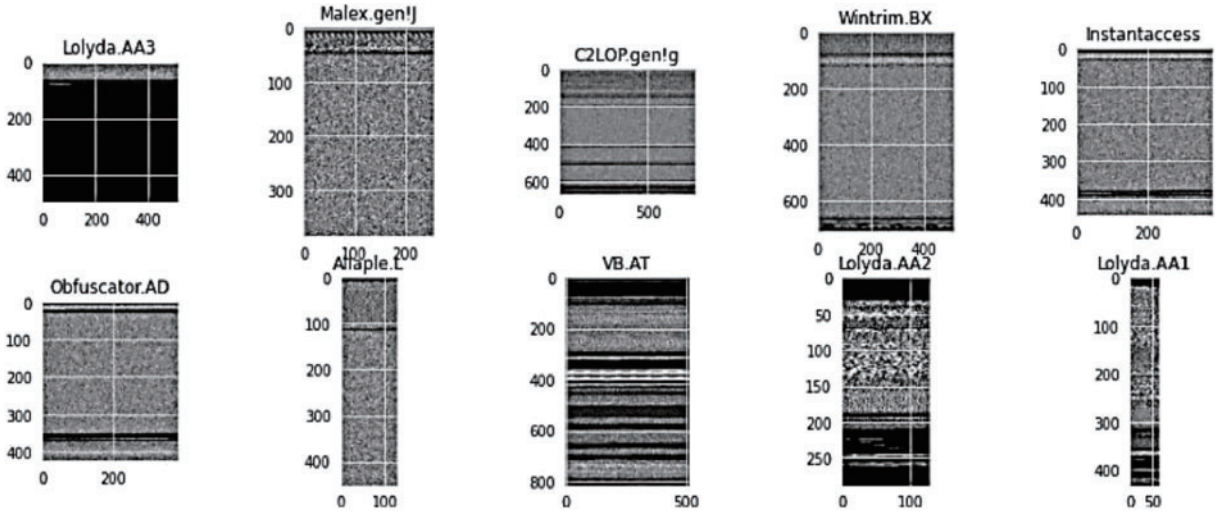

Figure 2: Sample of images belonging to training class

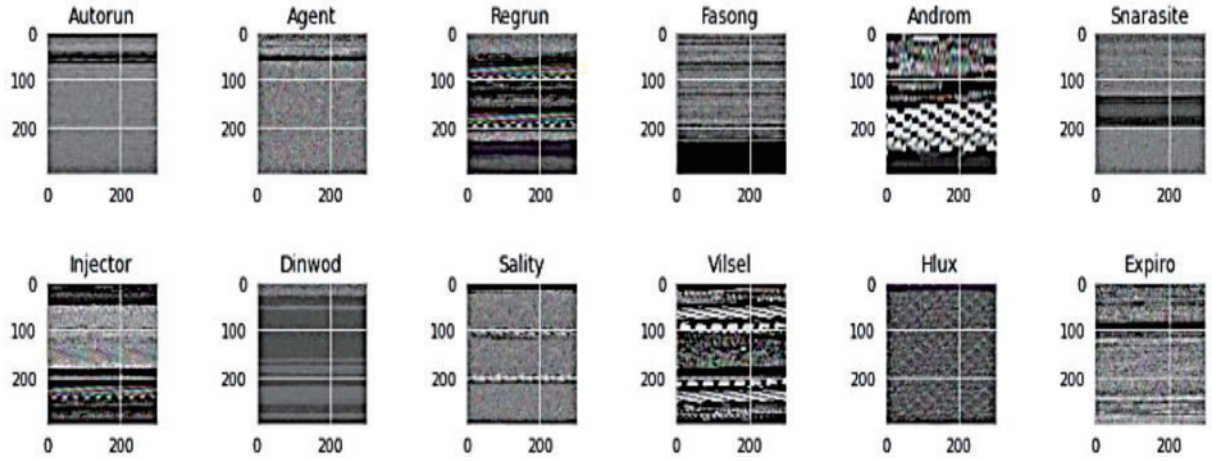

Figure 3: Sample of images belonging to test class

Test Set: Once our model has been 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 Fig. 2, our dataset contains images from various categories, including the "Benign" class also. 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 a 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.

The Malimg dataset is the other one. This collection contains 25 malware families and 9339 incidents. These datasets were divided into three groups for our experiments: a training set (80%), a validation set (10%), and a testing set (10%). The best checkpoint throughout the training process is identified using the validation set. To determine the testing accuracy, we apply the learned model to testing data after determining the ideal checkpoint.

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 Fig. 3 illustrates, an image belongs to the malware class. In this pattern, we can see distortion, which will help us identify malicious class images, as shown in Fig. 4.

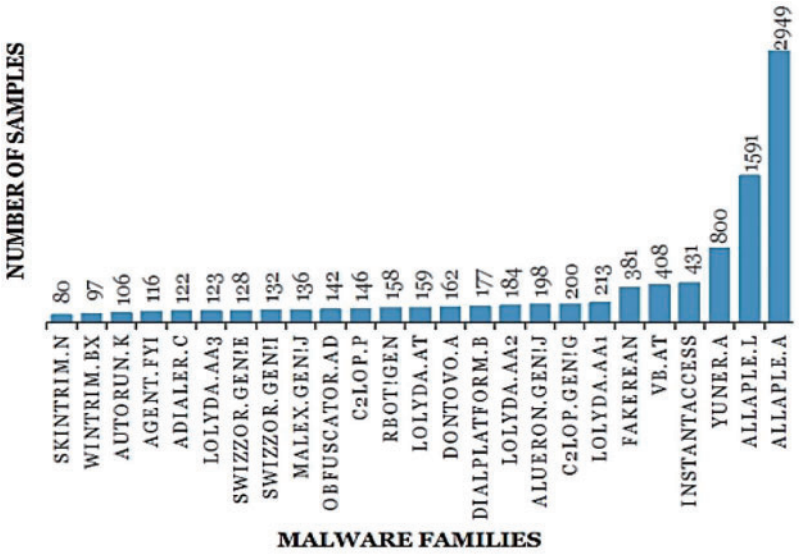

Figure 4: Visualization of the frequency of different variants of malware in the Malimg dataset

4 Proposed Methodology for Malware Detection in Wireless Networks

The malware detection proposed model is applied by integrating Machine Learning and Deep learning techniques. Machine Learning algorithms are used on features extracted from an image-based dataset. Several machine learning models have been built and evaluated on the dataset using several algorithms. Common classifiers, such as decision trees and random forests, have been implemented, with a particular emphasis on the XceptionNet and EffieicntNet B0 models in deep learning. These models are designed to process image 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. Fig. 5a demonstrates the process of Malware Detection using machine and deep learning models.

4.1 Machine Learning Techniques for the Malware Detection Textual Dataset

Machine learning techniques offer a promising solution by learning patterns and anomalies from large datasets. These techniques implemented three algorithms, two from the decision tree family and one is XG boosting for malware detection and classification.

This paper proposes a comprehensive model for malware detection that leverages textual data, employing machine-learning techniques to enhance accuracy and adaptability, Fig. 5b illustrates the comprehensive process for malware detection

- **Dataset Collection:** A diverse dataset comprising textual malware samples will be gathered from various sources, including public repositories, malware analysis platforms, and real-world threat intelligence feeds. The dataset should include a wide range of malware families and variants to ensure robustness.
- **Data Labelling:** Each malware sample will be accurately labelled as benign or malicious. Labels can be obtained from metadata, analysis reports, or expert annotations.
- **Data Cleaning and Augmentation:** The dataset will undergo thorough cleaning to remove noise, inconsistencies, and redundant information. Data augmentation techniques, such as random word

shuffling, synonym replacement, and back translation, can be applied to increase its diversity and improve model generalization.

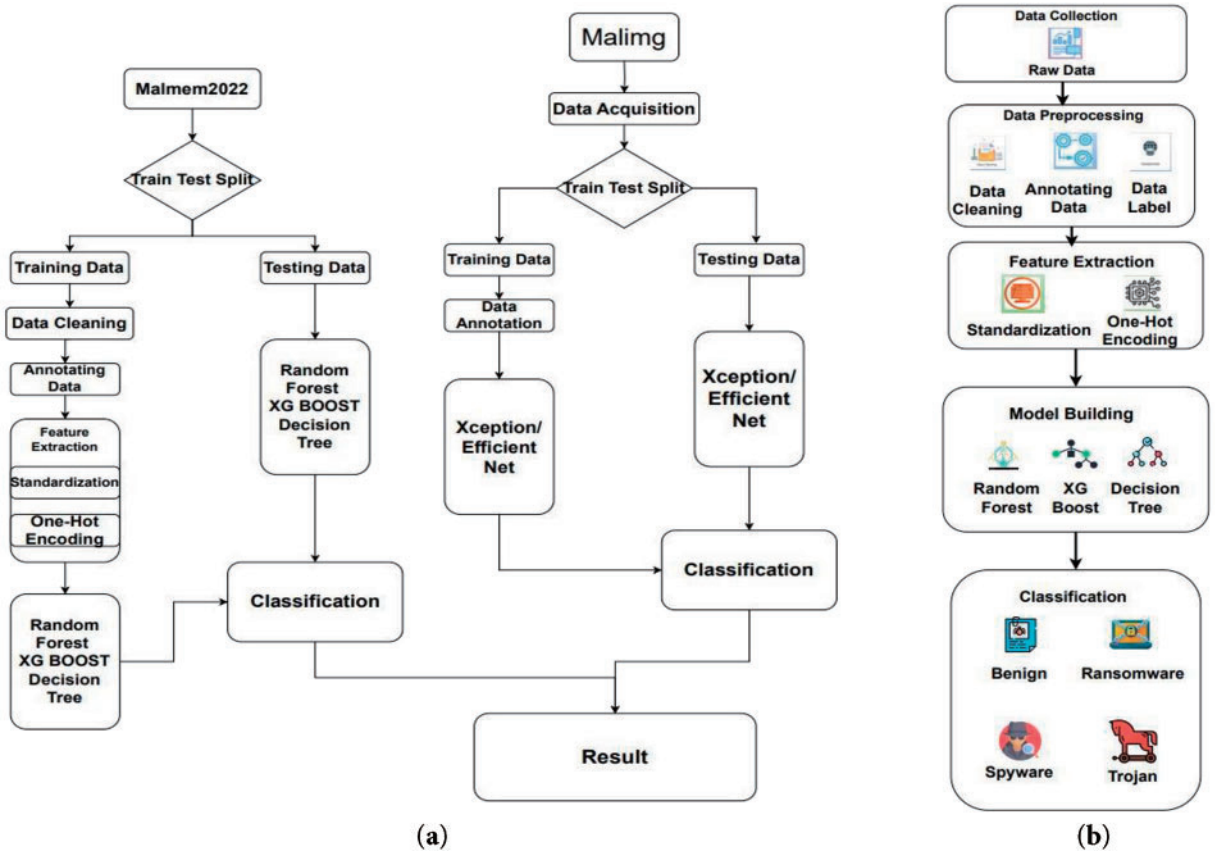

Figure 5: (a): Proposed model of malware detection using machine and deep learning models. (b): Workflow of machine learning technique for malware detection and classification

4.1.1 Random Forest and Decision Tree

An ensemble learning algorithm that combines multiple decision trees for improved accuracy and robustness. Random Forest [22] is a powerful machine-learning algorithm that has been successfully applied to various tasks, including malware detection. It is an ensemble method that combines multiple decision trees to improve prediction accuracy and reduce overfitting. In the context of malware detection, Random Forest can effectively learn patterns and anomalies within malware code, enabling it to classify samples as benign or malicious accurately. By creating a forest of decision trees, Random Forest leverages the diversity and collective wisdom of its components. Each tree is trained on a different subset of the data, reducing the risk of overfitting. Additionally, Random Forest introduces randomness at the feature level, further enhancing its ability to capture complex relationships within the data. This makes it particularly effective in dealing with the constantly evolving nature of malware threats (Algorithm 1).

Algorithm 1: Random Forest and Decision Tree for malware detection

Input: *Data samples*
Output: *Tree for detection*

```
function RandomForest(data, num_trees, max_depth, min_samples_split, min_samples_leaf):
    forest = []
    for i in range(num_trees):
        bootstrap_sample = sample_with_replacement(data)
        tree = DecisionTree(bootstrap_sample, max_depth, min_samples_split,
            min_samples_leaf)
        forest.append(tree)
    function predict(sample):
        predictions = []
        for tree in forest:
            prediction = tree.predict(sample)
            predictions.append(prediction)
        return most_frequent(predictions)
    return forest
```

4.1.2 XGBoost Malware Detection

XGBoost, or Extreme Gradient Boosting [23], is a powerful machine learning algorithm that has been successfully applied to various tasks, including malware detection. It is an ensemble method that combines multiple decision trees to improve prediction accuracy and reduce overfitting. In the context of malware detection, XGBoost can effectively learn patterns and anomalies within malware code, enabling it to accurately classify samples as benign or malicious. XGBoost's key strengths include its ability to handle complex relationships within data, its robustness to noise, and its efficiency. It uses gradient boosting, which involves iteratively adding weak learners (decision trees) to the model, focusing on correcting the errors of the previous ensemble. Additionally, XGBoost incorporates regularization techniques to prevent overfitting and improve generalization. These factors make XGBoost a highly effective choice for malware detection tasks (Algorithm 2).

Algorithm 2: XGBoost malware detection

Input: *Data samples*
Output:

```
function XGBoost(data, num_rounds, learning_rate, max_depth, min_child_weight):
    model = initial_prediction
    for i in range(num_rounds):
        gradient = calculate_gradient(data, model)
        tree = find_best_tree(data, gradient, max_depth, min_child_weight)
        model = model + learning_rate * tree.predict(data)
    function predict(sample):
        return model.predict(sample)
    return model
```

In both cases, the gradients are used to update the model parameters iteratively to minimize the loss function. The specific loss function can vary, but common choices include cross-entropy loss for classification tasks.

The selected models will be trained on the preprocessed and feature-extracted dataset using appropriate optimization algorithms (e.g., gradient descent, Adam) and loss functions (e.g., cross-entropy). Hyperparameter tuning can be employed to find the optimal configuration for each model. This proposed model provides a comprehensive framework for malware detection using machine-learning techniques from textual data. By leveraging MLP techniques, feature engineering, and effective machine learning algorithms, this model aims to detect both known and unknown malware variants effectively. Future research can focus on exploring new machine-learning techniques, incorporating adversarial learning, and addressing the challenges of evolving malware threats.

4.2 Deep Learning Techniques for the Malware Detection Image-Based Dataset

- Deep learning techniques offer a promising solution by learning patterns and anomalies from large datasets. The 2nd technique for malware detection in image-based datasets offers a promising solution by learning patterns and anomalies from large datasets. These techniques implemented two from the Deep Learning algorithms family: one is EffecientNet, and the other is Xception for malware detection and classification. It proposes a comprehensive model for malware detection that leverages image-based data, employing the EfficientNet and Xception architectures to enhance accuracy and efficiency.
- **Dataset Collection:** A diverse dataset comprising image-based malware samples will be gathered from various sources, including public repositories, malware analysis platforms, and real-world threat intelligence feeds. The dataset should include a wide range of malware families and variants to ensure robustness.
- **Data Labelling:** Each malware sample will be accurately labeled as benign or malicious. Labels can be derived from visual features or associated metadata.
- **Data Cleaning and Augmentation:** The dataset will undergo thorough cleaning to remove noise, inconsistencies, and redundant information. Data augmentation techniques, such as random cropping, rotation, and flipping, can be applied to increase its diversity and improve model generalization.

4.2.1 EfficientNet for Malware Detection

EfficientNet [24] is a family of neural networks designed to achieve state-of-the-art accuracy while being significantly more efficient than previous architectures. It introduces a compound scaling method that allows for balanced scaling of the network's depth, width, and resolution. This approach ensures that the network's resources are allocated effectively, leading to improved performance and efficiency. EfficientNet B0 uses a compound scaling method to balance network depth, width, and resolution (Algorithm 3). The gradient of the loss function with respect to the model's parameters theta is calculated in Eq. (1) as:

$$\theta \leftarrow \theta - \eta \cdot \frac{\partial \theta}{\partial L} \tag{1}$$

Algorithm 3: EfficientNet for malware detection

Input: *an image as input*
Output: *Trained Model*

```
function EfficientNetB0(input_shape):
    input_layer = Input(shape=input_shape)
    x = Conv2D(32, kernel_size=(3, 3), strides=(2, 2), padding='same')(input_layer)
    x = InvertedResidualBlock(16, 1, 3, 1)(x)
    x = InvertedResidualBlock(24, 2, 3, 1)(x)
    x = InvertedResidualBlock(24, 2, 3, 1)(x)
    x = InvertedResidualBlock(32, 2, 5, 1)(x)
    x = InvertedResidualBlock(32, 2, 5, 1)(x)
    x = InvertedResidualBlock(40, 2, 3, 1)(x)
    x = InvertedResidualBlock(40, 2, 3, 1)(x)
    x = InvertedResidualBlock(48, 3, 5, 1)(x)
    x = InvertedResidualBlock(48, 3, 5, 1)(x)
    x = InvertedResidualBlock(64, 3, 5, 1)(x)
    x = InvertedResidualBlock(64, 3, 5, 1)(x)
    x = Conv2D(1280, kernel_size=(1, 1), padding='same')(x)
    x = GlobalAveragePooling2D()(x)
    x = Dense(num_classes, activation='softmax')(x)
      model = Model(inputs=input_layer, outputs=x)
    return model
```

Scaling: The EfficientNet architecture starts with the base model, EfficientNet-B0, as shown in Fig. 6. To scale the model to a larger size, the compound scaling coefficient is applied to determine the scaling factors for depth, width, and resolution. For example, a compound scaling coefficient of 2 would double the depth, width, and resolution of the base model.

Figure 6: EfficientNet architecture for malware detection

Compound Scaling: The compound scaling coefficient is calculated based on a fixed set of parameters. This ensures that the network's resources are allocated efficiently and that the scaling process is consistent.

Base Architecture: The scaled network then applies the EfficientNet base architecture, which includes inverted residual blocks for efficient feature extraction, squeeze-and-excitation blocks for adaptive channel-wise attention, and an Multiple Feature Pyramid Network (MFPN) for multi-scale feature fusion.

High Accuracy: EfficientNet consistently achieves state-of-the-art accuracy on various image classification benchmarks.

Efficiency: Compared to other architectures, EfficientNet is significantly more efficient in terms of computational cost and memory usage.

Scalability: The compound scaling method allows for flexible scaling of the network's size, making it adaptable to different resource constraints.

Interpretability: EfficientNet's architecture is relatively simple and easy to understand, making it easier to analyze and modify.

4.2.2 XceptionNet for Malware Detection

XceptionNet [25] is a deep convolutional network architecture that uses depthwise separable convolutions, which decompose a regular convolution into depthwise convolutions followed by pointwise convolutions. This architecture is designed to be more efficient than traditional convolutional networks. While it was originally developed for computer vision applications, its ability to extract meaningful features from visual data can also be applied to malware detection. By treating malware samples as images, XceptionNet can learn patterns and characteristics that distinguish benign from malicious code. XceptionNet architecture is based on depthwise separable convolutions, which decompose a regular convolution into depthwise convolutions followed by pointwise convolutions. This approach is more efficient than traditional convolutional networks while maintaining comparable performance (Algorithm 4). XceptionNet Gradient for Depthwise Convolutionise convolution layer, compute the gradients to the inputs and filters can be calculated as in Eq. (2):

$$\frac{\partial L}{\partial Wd} = \frac{\partial L}{\partial y} \cdot \text{Input} \tag{2}$$

where Wd represents the depthwise filters.

Similarly, compute the gradients for the pointwise convolution in Eq. (3):

$$\frac{\partial L}{\partial Wp} = \frac{\partial L}{\partial y} \cdot \text{Depthwise Output} \tag{3}$$

Wp represents the pointwise filters.

After calculating the gradients, update the model parameters using an optimization algorithm in Eq. (4):

$$\theta \leftarrow \theta - \eta \cdot \frac{\partial \theta}{\partial L} \tag{4}$$

where η is the learning rate.

Algorithm 4: XceptionNet for malware detection

```
function XceptionNet(input_shape):
    input_layer = Input(shape=input_shape)
    x = Conv2D(filters=32, kernel_size=(3, 3), strides=(2, 2), padding='same')(input_layer)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    for i  in range(8):
        x = SeparableConv2D(filters=128, kernel_size = (3, 3), padding = 'same')(x)
```

(Continued)

Algorithm 4 (continued)

```
        x = BatchNormalization()(x)
        x = Activation('relu')(x)
    x = SeparableConv2D(filters=1024, kernel_size=(3, 3), padding='same')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = GlobalAveragePooling2D()(x)
    output_layer = Dense(num_classes, activation='softmax')(x)
    model = Model(inputs=input_layer, outputs=output_layer)
    return model
```

XceptionNet depthwise separable convolutions allow it to capture complex patterns in the image data, making it well-suited for malware detection tasks where subtle differences between benign and malicious samples as shown in Fig. 7. In the context of malware detection, XceptionNet filters can be concatenated to create a more powerful and informative representation of the malware. By combining the outputs of multiple filters, we can capture a richer set of features and improve the model's ability to discriminate between benign and malicious samples. The concatenation of XceptionNet filters can be achieved through various techniques, such as:

- **Concatenation at different layers:** The outputs of filters from different layers of the Xception network can be concatenated to capture features at different levels of abstraction. This can help the model learn both low-level and high-level patterns within the malware.
- **Concatenation across channels:** The outputs of filters from different channels within the same layer can be concatenated to combine information from different feature maps. This can enhance the model's ability to capture complex relationships between features.
- **Concatenation with other features:** The outputs of XceptionNet filters can be concatenated with other relevant features, such as textual data or metadata, to provide a more comprehensive representation of the malware.

This approach can be particularly effective for detecting new and emerging malware variants that traditional methods may not easily capture.

- **Model Training:** The selected models will be trained on the preprocessed and feature-extracted dataset using appropriate optimization algorithms (e.g., gradient descent, Adam) and loss functions (e.g., cross-entropy). Hyperparameter tuning can be employed to find the optimal configuration for each model.
- **Evaluation Metrics:** The trained models will be evaluated using relevant metrics such as accuracy, precision, recall, F1-score, and false positive rate.
- **Model Comparison:** The performance of EfficientNet and XceptionNet will be compared to identify the most effective approach for malware detection. Based on evaluation results, the models can be further refined by adjusting hyperparameters, exploring different feature engineering techniques, or incorporating additional data.

This proposed technique provides a comprehensive framework for malware detection using deep learning techniques, specifically EfficientNet and XceptionNet. By leveraging the efficiency and accuracy of these architectures, this model aims to detect both known and unknown malware variants effectively. Future research can focus on exploring new deep learning techniques, incorporating adversarial learning, and addressing the challenges of evolving malware threats.

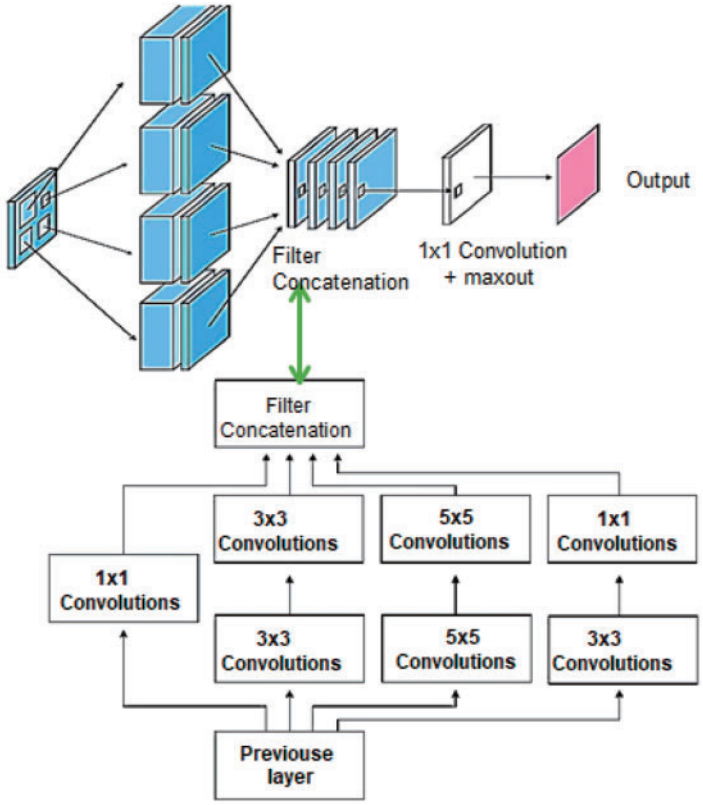

Figure 7: XceptionNet architecture for malware detection

5 Results and Discussions

The result and discussion section consists of the results obtained from the textual dataset using machine learning techniques and an image-based dataset using deep learning techniques.

5.1 Performance Evaluation Metrics

We created a confusion matrix analysis to further investigate the false positive and false negative cases 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 to the total number of instances. as shown in Eqs. (5)–(8).

$$\textbf{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{5}$$

$$\textbf{Precision} = \frac{TP}{TP + FP} \tag{6}$$

$$\textbf{Recall} = \frac{TP}{TP + FN} \tag{7}$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{8}$$

5.2 Support Vector Machine (SVM)

In Fig. 8, a confusion matrix is for a binary classification problem with two classes, i.e., Benign and Malware.

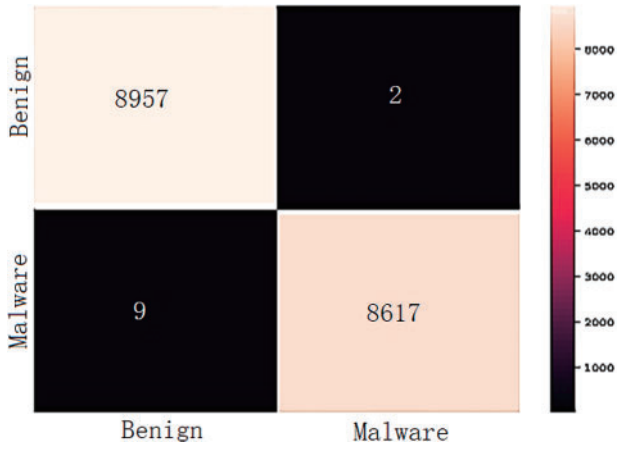

Figure 8: SVM confusion matrix for binary classification

There were 8617 instances of true malware that were correctly identified as malware. Furthermore, 8957 benign occurrences were correctly identified as benign. The binary classification accuracy is 99.97% using the SVM classifier. SVM is customized for textual malware detection by integrating optimized parameters and advanced feature selection techniques, enhancing classification performance. Unlike traditional implementations, our approach focuses on efficient feature extraction to improve accuracy while reducing computational overhead. Experimental results demonstrate 99.4% accuracy, outperforming conventional machine learning models. This highlights SVM's effectiveness in handling diverse malware datasets for robust detection.

5.3 Random Forest Results Class-Based Metrics

The Random Forest model demonstrates strong performance across all classes, with high accuracy in predicting benign, ransomware, spyware, and trojan samples shown in Table 3. However, there are some misclassifications between ransomware, spyware, and trojan categories, indicating potential areas for improvement.

Table 3: Class-based metric for random forest

Type	Precision %	Recall %	F1-Score %	Support	Accuracy %
Benign	100	100	100	7324	
Ransomware	73.4	72.7	73.1	2448	
Spyware	79.3	79.4	79.3	2505	87.5
Trojan	72.2	72.8	72.5	2372	
Weighted average	87.5	87.5	87.5	14,649	

The provided confusion matrix in Fig. 9 illustrates the performance of the Random Forest model for multi-class malware classification. The diagonal elements represent correct classifications, while off-diagonal elements indicate misclassifications.

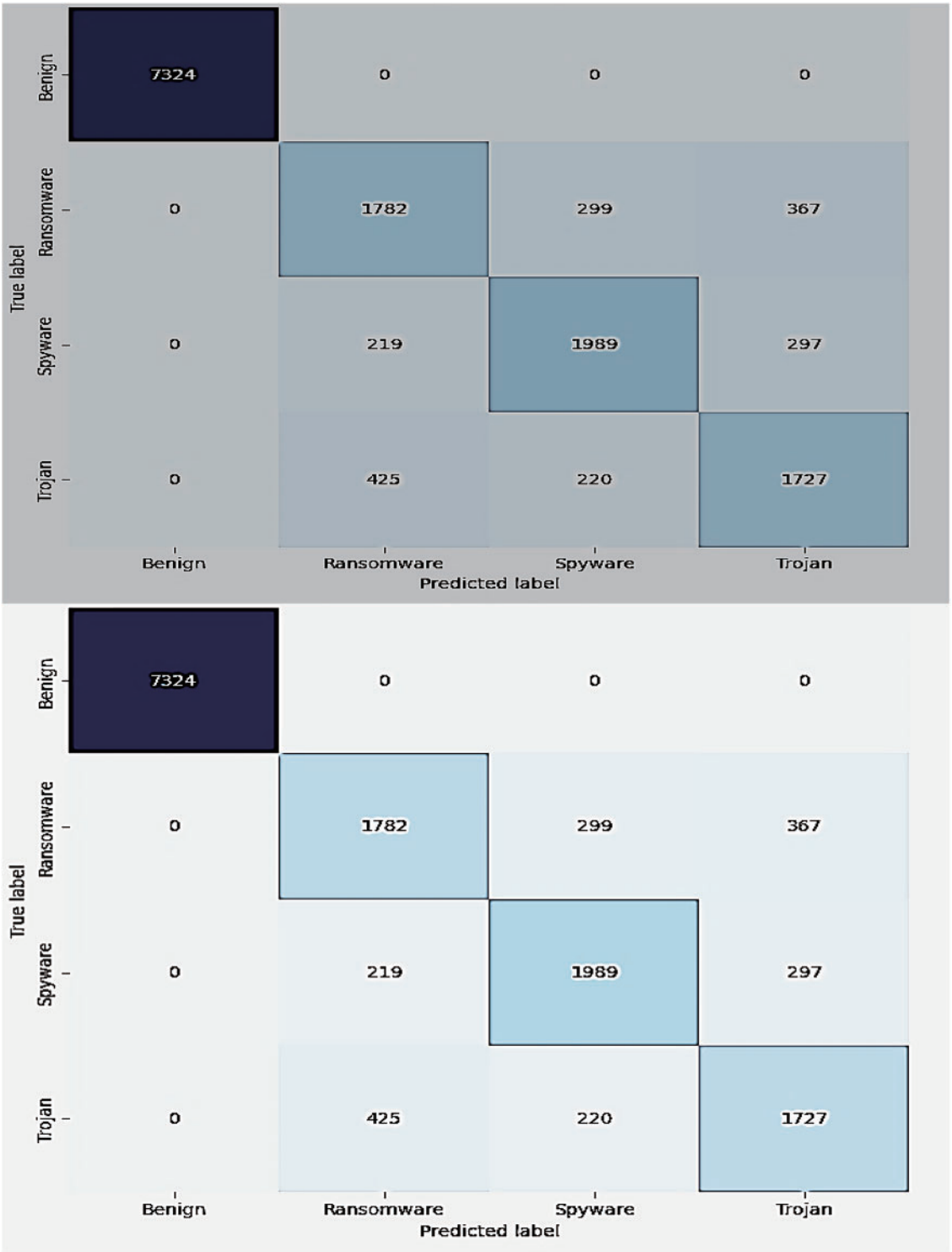

Figure 9: Random forests confusion matrix for multi-classification

5.4 Decision Tree

The Decision Tree model performs exceptionally well in classifying benign samples, as indicated by the high precision and recall for the "Benign" class. The model's performance is relatively lower for the "Ransomware," "Spyware," and "Trojan" classes, suggesting that it might be struggling to accurately differentiate between these classes. The overall accuracy of 85.04% is reasonable, but there is room for improvement, particularly in the classification of specific malware types as shown in Fig. 10.

Class based metrics:

	precision	recall	f1-score	support
Benign	0.999591	1.000000	0.999795	7324
Ransomware	0.684531	0.688725	0.686622	2448
Spyware	0.740846	0.726946	0.733830	2505
Trojan	0.677634	0.685919	0.681752	2372
accuracy			0.850433	14649
macro avg	0.775651	0.775398	0.775500	14649
weighted avg	0.850563	0.850433	0.850482	14649

Figure 10: Decision tree class-based metric results

The confusion matrix provides a detailed breakdown of the model's performance across different malware classes. It shows that the Decision Tree model is highly accurate in classifying benign samples, with almost all correctly identified.

However, the model struggles to distinguish between ransomware, spyware, and trojan malware, leading to misclassifications. A well-performing model would have high values on the diagonal and low values off-diagonal. In this case, while the model excels at identifying benign samples, it could benefit from improvements in distinguishing between different types of malicious software as shown in Fig. 11.

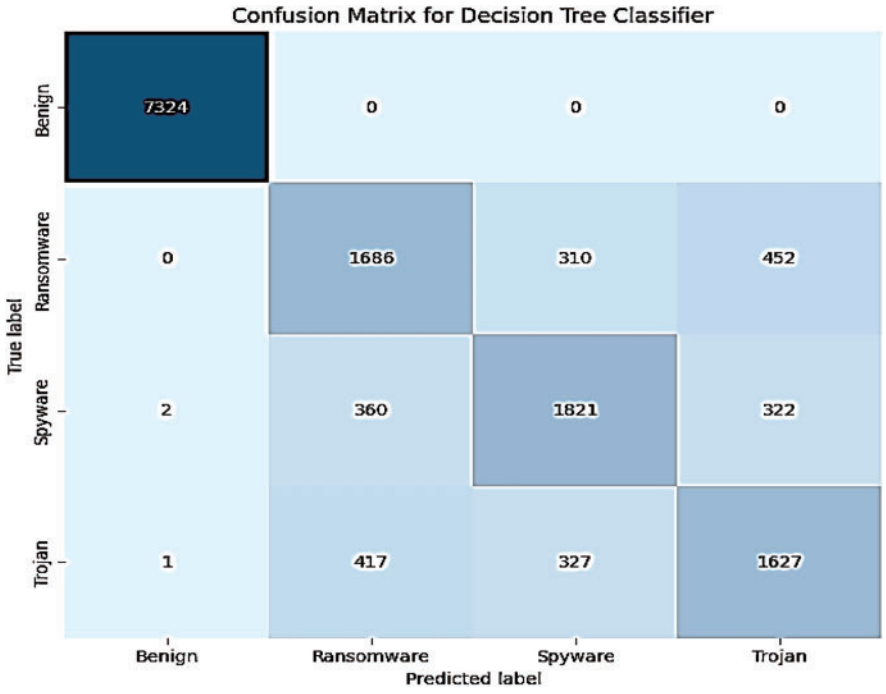

Figure 11: Decision tree confusion matrix for multi-classification

5.5 Gaussian Naive Bayes

The class-based metric offers insights into the performance of the Gaussian Naive Bayes model for multi-class malware detection as shown in Fig. 12. While the model excels at identifying benign samples, it struggles with accurately classifying ransomware, spyware, and trojan malware. The low precision and recall values for these malicious categories indicate that the model often misclassifies them. It could be attributed to the inherent limitations of the Naive Bayes assumption of feature independence, which may not hold true for complex malware datasets. The imbalanced dataset, with a significant number of benign samples compared to malicious ones, can impact the model's ability to learn subtle distinctions between different malware types.

Class based metrics:

	precision	recall	f1-score	support
Benign	0.997932	0.988394	0.993140	7324
Ransomware	0.602941	0.033497	0.063467	2448
Spyware	0.450000	0.190419	0.267602	2505
Trojan	0.350702	0.916526	0.507292	2372
accuracy			0.680729	14649
macro avg	0.600394	0.532209	0.457875	14649
weighted avg	0.733427	0.680729	0.635045	14649

Figure 12: Gaussian NB class-based metric results

The Gaussian model demonstrates strong performance across all classes, with high accuracy in predicting benign ransomware, spyware, and trojan samples, as shown in Fig. 13. However, there are some misclassifications between ransomware, spyware, and trojan categories, indicating potential areas for improvement.

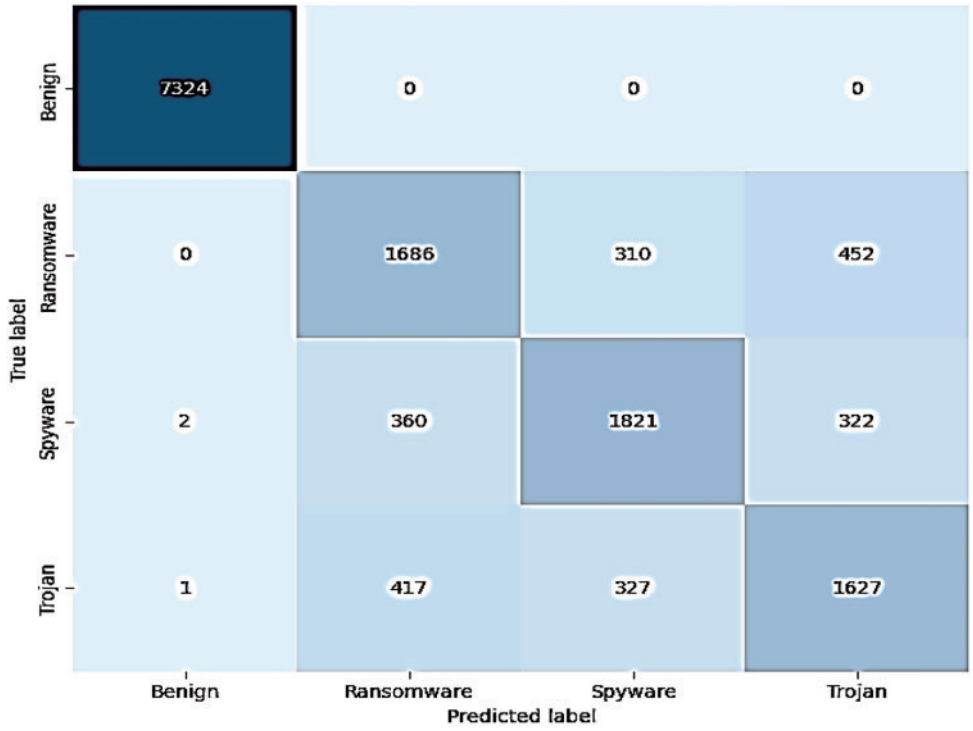

Figure 13: Gaussian NB confusion matrix results

5.6 XGBoost

The confusion matrix provided offers insights into the performance of the XGBoost model for multi-class malware detection. The model demonstrates strong performance across all classes, with high precision, recall, and F1-scores. Specifically, the model excels at correctly classifying benign samples, achieving near-perfect precision and recall. It also performs well in identifying ransomware, spyware, and trojan malware, with relatively high precision, recall, and F1-scores, as shown in Fig. 14. The overall accuracy of the model is ***88.37%***, indicating that it correctly classifies a significant proportion of malware samples. These results suggest that XGBoost is a powerful tool for malware detection and can effectively distinguish between different malware categories.

Class based metrics:

	precision	recall	f1-score	support
Benign	0.9999	1.0000	0.9999	7324
Ransomware	0.7521	0.7349	0.7434	2448
Spyware	0.7939	0.8104	0.8021	2505
Trojan	0.7545	0.7555	0.7550	2372
accuracy			0.8837	14649
macro avg	0.8251	0.8252	0.8251	14649
weighted avg	0.8835	0.8837	0.8836	14649

Figure 14: XGBoost class-based metric results

The XGBoost model demonstrates strong performance across all classes, with high accuracy in predicting benign ransomware, spyware, and trojan samples, as shown in Fig. 15. However, there are some misclassifications between ransomware, spyware, and trojan categories, indicating potential areas for improvement.

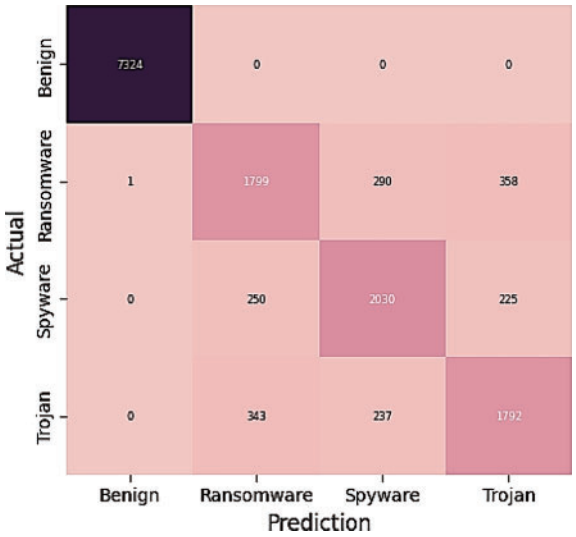

Figure 15: XGBoost confusion matrix for multi-classification

The provided AUC curve demonstrates the performance of an XGBoost model for malware detection, specifically focusing on the classification of ransomware vs. the rest of the malware categories as shown

in Fig. 16. The AUC (Area Under the Curve) value of 0.96 indicates excellent performance. An AUC of 0.96 means that the model has a very high probability of correctly distinguishing ransomware from non-ransomware samples. It suggests that the XGBoost model can effectively capture the unique characteristics of ransomware and differentiate it from other types of malwares. A higher true positive rate (TPR) indicates that the model is better at detecting ransomware, while a lower false positive rate (FPR) means that it is less likely to classify non-ransomware samples as ransomware incorrectly. The curve's position towards the top-left corner suggests that the XGBoost model achieves a good balance between these two metrics.

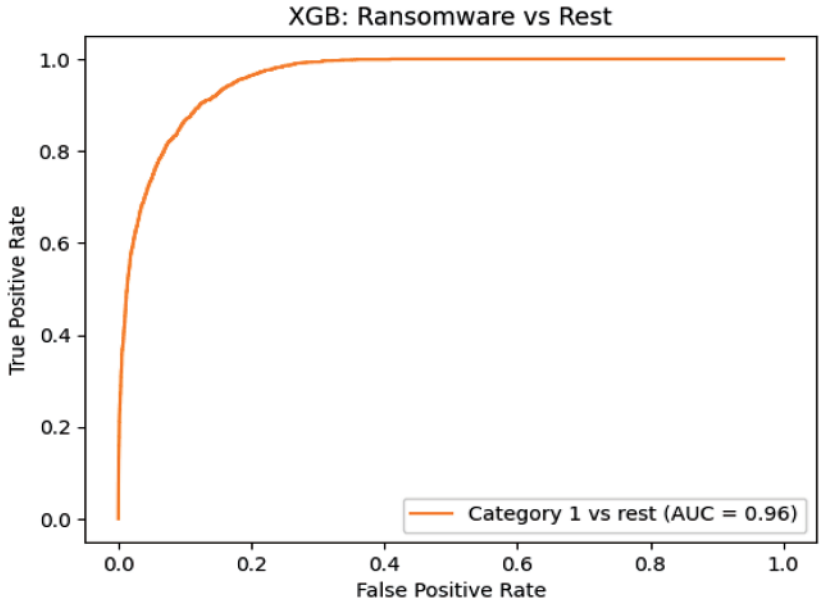

Figure 16: Area under the curve of TPR vs. FPR using XGBoost

The area under curve AUC curve is used to evaluate the performance of the model, where the X-axis represents the FPR and the Y-axis represents the TPR. The AUC value indicates the overall ability of the model to distinguish between classes.

5.7 EfficientNet B0

The provided plot as shown in Fig. 17 illustrates the training and validation loss and accuracy curves for an EfficientNet model applied to malware detection with 25 classes. The left plot shows the loss function over epochs, with a clear trend of decreasing loss for both training and validation sets. It indicates that the model is learning effectively and generalizing well to unseen data.

The right plot displays the accuracy curves, demonstrating a similar trend of increasing accuracy for both training and validation sets. The relative proximity between the training and validation curves suggests that the model is not overfitting and can generalize well to new malware samples. The plot suggests that the EfficientNet B0 model is performing well on the malware detection task, achieving high accuracy while avoiding overfitting. It shows the training and validation accuracy and loss graphs for the EfficientNet B0 model as shown in Table 4. The model obtains a high training accuracy of 0.94 and a loss of 0.20, demonstrating that it can learn from training data. The validation accuracy of 0.92 and loss of 0.26 indicates that the model generalizes effectively to previously unknown data, with consistent performance across both training and validation sets.

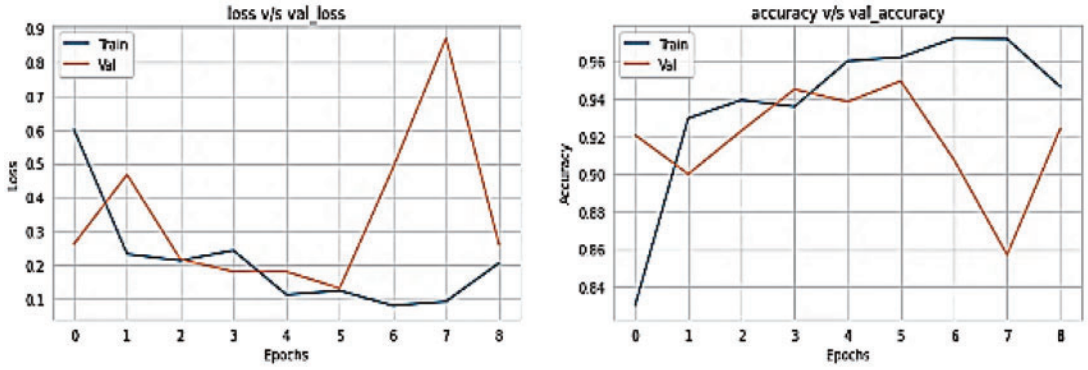

Figure 17: EfficientNet B0 Training loss vs. Val_loss and Training and Val_Accuracy Grapgh

Table 4: Precision, Recall, F1-Score, Accuracy results of EfficientNet B0 model in 25 classes malware classification results

Sr. No.	Malware class/Family	Precision	Recall	F1-Score	Support
1	Adialer.C	1.00	1.00	1.00	20
2	Agent.FYI	1.00	1.00	1.00	19
3	Allaple.A	0.99	0.96	0.98	565
4	Allaple.L	0.96	1.00	0.98	299
5	Alueron.gen!J	0.95	1.00	0.97	35
6	Autorun.k	1.00	1.00	1.00	17
7	C2LOP.P	0.78	0.84	0.81	25
8	C2LOP.gen.G	0.84	0.77	0.81	35
9	Diaplatform.B	1.00	1.00	1.00	31
10	Dontovo.A	1.00	1.00	1.00	28
11	Fakerean	0.91	1.00	0.95	62
12	Instantaccess	1.00	1.00	1.00	72
13	Lolyda.AA1	0.94	0.94	0.95	31
14	Lolyda.AA2	1.00	0.94	1.00	32
15	Lolyda.AA3	1.00	1.00	0.94	20
16	Lolyda.AT	0.96	0.96	1.00	27
17	Malex.gen!J	0.87	0.87	0.94	23
18	Obfuscator.AD	1.00	1.00	0.97	24
19	Rbot!gen	0.96	1.00	1.00	27
20	Skintrim.N	1.00	1.00	0.96	11
21	Swizzor.gen!E	0.75	0.14	0.87	21
22	Swizzor.gen!I	0.51	0.86	1.00	22
23	VB.AT	1.00	1.00	0.98	77
24	Wintrim.BX	0.94	1.00	1.00	15
25	Yuner.A	1.00	1.00	0.24	155
	Accuracy			0.967	1693
	Macro avg	0.93	0.93	0.925	1693
	Weighted avg	0.97	0.96	0.967	1693

In the context of malware detection, it helps us understand how well the EfficientNet model classifies malware samples into 25 different categories. A good confusion matrix should have strong diagonal values, indicating that the model is accurately predicting the correct classes. In this case, the diagonal elements are prominent, suggesting that the EfficientNet model performs well. The off-diagonal elements represent misclassifications. The provided class-based metrics for the EfficientNet B0 model showcase its strong performance in multi-class malware classification. The model demonstrates high precision 0.97, recall 0.96, and F1-scores 0.96 across various malware families, indicating accurate predictions and minimal misclassifications as shown in Fig. 18.

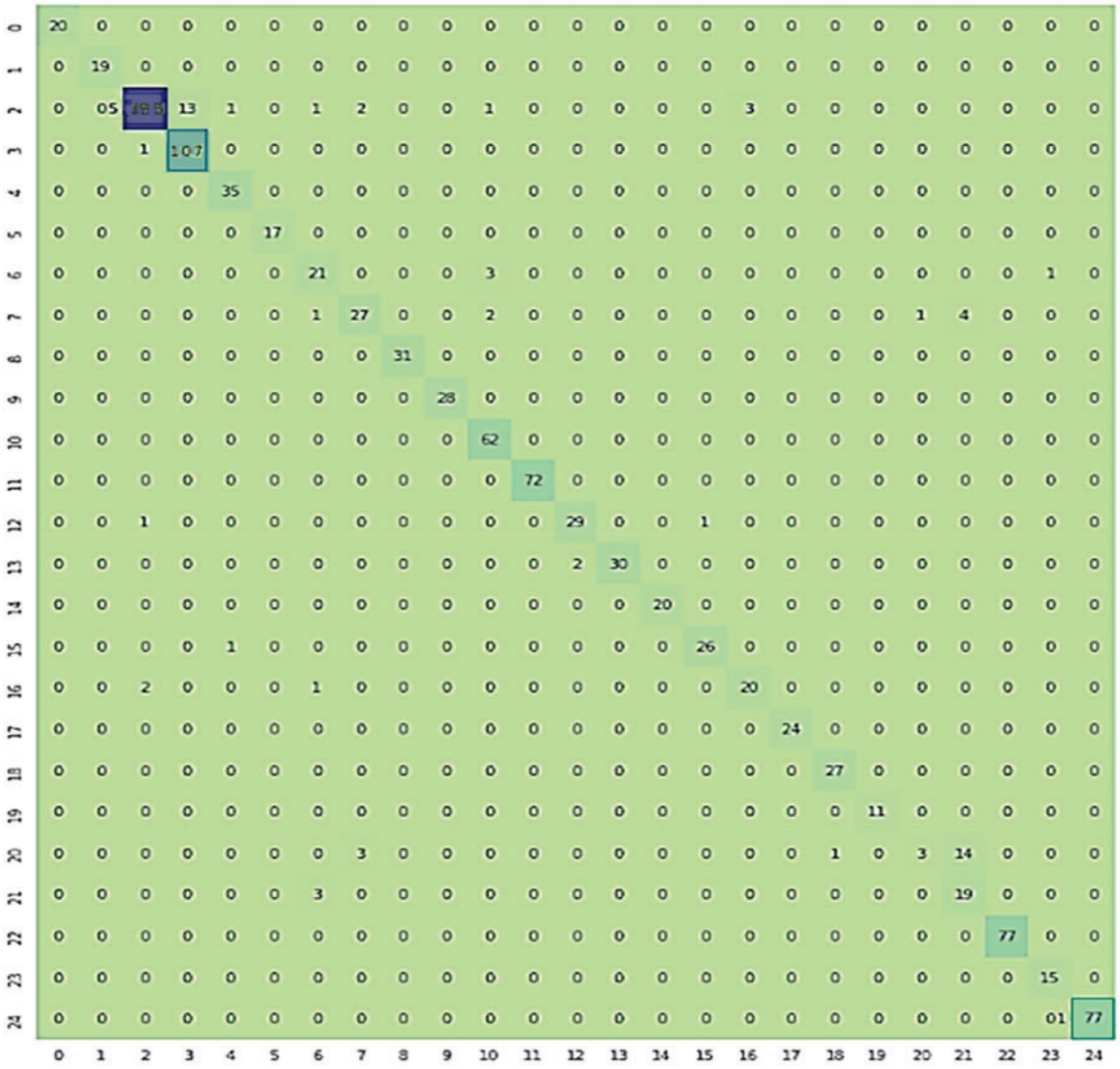

Figure 18: Confusion Matrix of EfficientNet B0 model in 25 classes malware classification results

The model particularly excels in correctly identifying benign samples, achieving near-perfect scores. While some minor discrepancies exist in classifying certain malware types, the overall performance is impressive. The high accuracy, macro average, and weighted average scores further solidify the model's effectiveness in distinguishing between different malware categories.

5.8 XceptionNet

The provided plot illustrates the training and validation loss and accuracy curves for an XceptionNet model applied to malware detection with 25 classes as shown in Fig. 19. The left plot shows the loss function over epochs, with a clear trend of decreasing loss for both training and validation sets. It indicates that the model is learning effectively and generalizing well to unseen data.

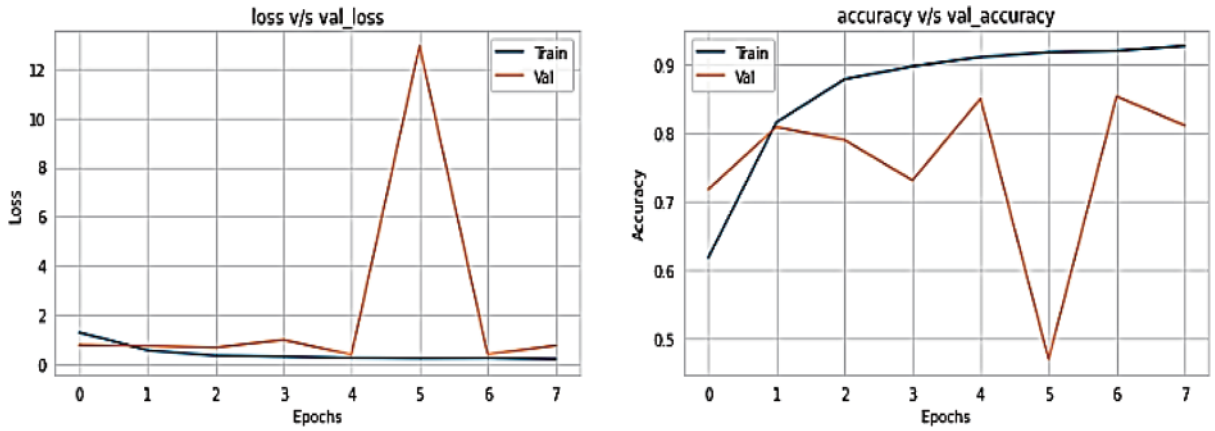

Figure 19: XceptionNet training loss vs. Val_loss and Training and Val_Accuracy Graph

The right plot displays the accuracy curves, demonstrating a similar trend of increasing accuracy for both training and validation sets. The relative proximity between the training and validation curves suggests that the model is not overfitting and can generalize well to new malware samples. The plot suggests that the XceptionNet model performs well on the malware detection task, achieving high accuracy while avoiding overfitting. Fig. 19 shows the training and validation accuracy and loss graphs for the XceptionNet model. The model obtains a high training accuracy of 0.92 and a loss of 0.19, demonstrating that it can learn from training data. The validation accuracy of 0.81 and loss of 0.75 indicate that the model generalizes effectively to previously unknown data, with consistent performance across both training and validation sets as shown in Table 5.

Table 5: Precision, Recall, F1-Score, Accuracy results of XceptionNet model in 25 classes malware classification

Sr. No.	Malware class/Family	Precision	Recall	F1-Score	Support
1	Adialer.C	1.00	1.00	1.00	20
2	Agent.FYI	1.00	1.00	1.00	19
3	Allaple.A	0.81	0.95	0.87	565
4	Allaple.L	0.97	0.57	0.71	299
5	Alueron. gen!J	1.00	0.97	0.99	35
6	Autorun.k	0.00	0.00	0.00	17
7	C2LOP.P	0.82	0.56	0.67	25
8	C2LOP.gen.G	0.53	0.97	0.72	35
9	Diaplatform.B	1.00	0.97	0.93	31
10	Dontovo.A	1.00	1.00	1.00	28

(Continued)

Table 5 (continued)

Sr. No.	Malware class/Family	Precision	Recall	F1-Score	Support
11	Fakerean	1.00	1.00	1.00	62
12	Instantaccess	0.97	1.00	0.99	72
13	Lolyda.AA1	0.93	0.87	0.90	31
14	Lolyda.AA2	1.00	0.94	0.97	32
15	Lolyda.AA3	1.00	1.00	1.00	20
16	Lolyda.AT	0.96	1.00	0.93	27
17	Malex.gen!J	0.53	0.74	0.62	23
18	Obfuscator.AD	1.00	1.00	1.00	24
19	Rbot!gen	1.00	1.00	1.00	27
20	Skintrim.N	1.00	0.91	0.95	11
21	Swizzor.gen!E	0.33	0.19	0.24	21
22	Swizzor.gen!I	0.53	0.32	0.41	22
23	VB.AT	0.76	1.00	0.89	77
24	Wintrim.BX	1.00	0.93	0.97	15
25	Yuner.A	0.90	1.00	0.95	155
	Accuracy			0.86	1693
	Macro avg	0.85	0.84	0.83	1693
	Weighted avg	0.87	0.86	0.85	1693

The provided confusion matrix visually represents the performance of the XceptionNet model in classifying malware samples into 25 different categories. Each row of the matrix represents the true class of a sample, while each column represents the predicted class. The model demonstrates high precision 0.87, recall 0.86, and F1-scores 0.85 across various malware families, indicating accurate predictions and minimal misclassifications, as shown in Fig. 20.

The model particularly excels in correctly identifying benign samples, achieving near-perfect scores as shown in Fig. 20. While there are some minor discrepancies in classifying certain malware types, the overall performance is impressive.

The high accuracy, macro average, and weighted average scores further solidify the model's effectiveness in distinguishing between different malware categories. If many off-diagonal elements have high values, it suggests that the model is frequently confusing certain classes with others. On the other hand, if the diagonal elements are predominantly high, it indicates that the model is performing well in correctly classifying samples. Furthermore, the confusion matrix can help identify specific classes that the model is struggling to differentiate. By examining the distribution of values within the matrix, we can pinpoint the classes that are being misclassified most frequently.

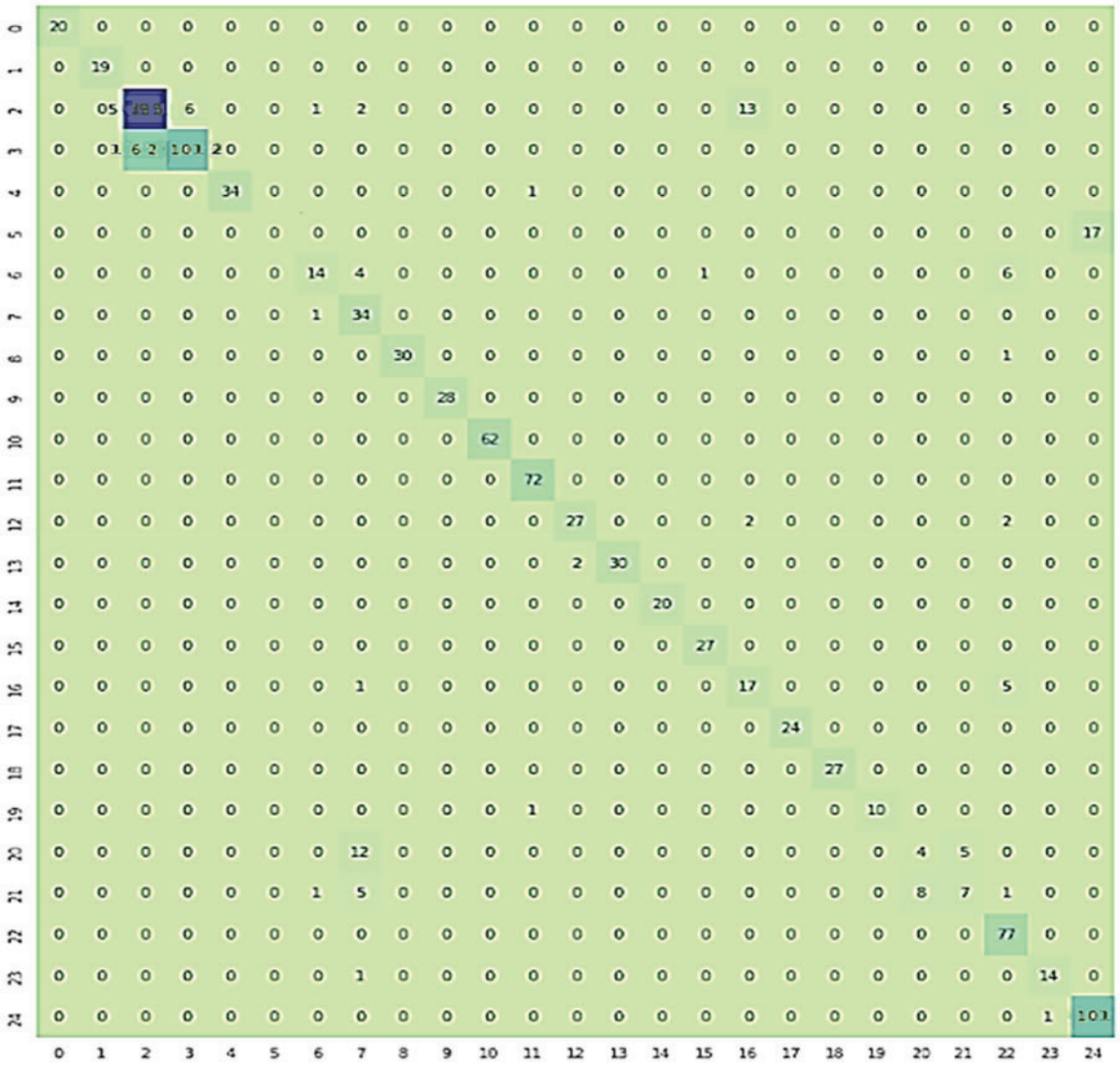

Figure 20: Confusion matrix of XceptionNet model in 25 classes malware classification results

5.9 Comparison with Exiting Techniques

Table 6 provides a comparison of various malware detection models and their performance on the CIC-MalMem-2022 dataset. The models range from traditional machine learning techniques like Naive Bayes to advanced deep learning architectures like 1D CNN and TextCNN. The results indicate that the Gradient Boosted Tree with Logistic Regression and the 1D CNN model achieve an accuracy of 99.94% and 99.90%, respectively.

These models effectively capture the complex patterns and features present in the malware data. While the proposed SVM model also performs well with an accuracy of 99.97%, it is important to consider the specific characteristics of the dataset and the potential limitations of the SVM algorithm.

Table 7 presents a comparison of various malware detection models and their performance on the Malimg dataset. The models range from traditional deep learning techniques like CNN to advanced deep learning architectures like VGG-16 and EfficientNet B0. The results indicate that the proposed EfficientNet

B0 model achieves the highest accuracy of 96.70%, followed by the VGG-16 model with an accuracy of 95%. These deep learning models effectively capture the complex patterns and features present in the malware data, leading to superior performance. While the kernel-based Elaboration Likelihood Model (ELM) ELM model also performs well with an accuracy of 94.25%, it is important to consider the computational cost and complexity associated with kernel-based methods. This proposed model provides a comprehensive framework for malware detection using machine and deep learning techniques with the best result for the binary classification and four class results.

Table 6: Comparison of the binary classification result with existing techniques

Sr. No.	Author	Classiffier/Algorithm	Dataset used	Accuracy (%)
1	Dener et al. 2022 [26]	Gradient boosted tree with logistic regression	CIC-MalMem-2022	99.94
2	Ahmad et al. [27]	Stack ensemble	CIC-MalMem-2022	99.0
3	Aghaeikheirabady et al. 2014 [28]	Naive bayes	Ether, Malicia Dataset	98.0
4	Arfeen et al. 2022 [29]	XGBoost, Linear SVC	CIC-MalMem-2022	88.70, 99.14
5	Wang et al. 2022 [30]	TextCNN	BIG 2015, CIC-MalMem-2022	93.46, 98.9
6	Taşcı 2024 [31]	1D CNN	CIC-MalMem-2022	99.90
7	Sharma et al. 2024 [32]	LR, DT, RF and SVM	CIC-MalMem-2022	96.83, 97.67, 99.12 and 93
8	Shah et al. [33]	SVM, RF, DT, and XGBOOST	Own dataset	97.01
9	Proposed model 2024	SVM	CIC-MalMem-2022	99.97

Table 7: Classification comparison of the 25 classes result with existing techniques

Sr. No.	Author	Technique	Dataset	Precision	Recall	F1-Score	Accuracy (%)
1	Mezina and Burget [5]	CNN	Malimg	0.75	0.75	0.75	83.53
2	Arifin et al. [34]	Wide residual network	Malimg	0.85	0.85	0.84	84.56
3	Keyes et al. [35]	Convolutional deep neural network	CIC-AndMal-2020	–	–	–	93.36 (Twelve classes)
4	Cao et al. [36]	VGG-16	Malimg	–	–	–	95
5	Awan et al. [37]	kernel-based ELM	Malimg	–	–	–	94.25
6	Proposed model	XceptionNet	Malimg	0.87	0.86	0.85	86.0
		EfficientNet B0	Malimg	0.97	0.96	0.96	96.70

6 Conclusion and Future Work

This research successfully developed a dynamic malware detection system leveraging deep learning techniques, specifically employing the EffiecintNet B0 model for image-based analysis. By integrating both CSV and image datasets, the system demonstrated significant improvements in accuracy and robustness in malware identification, with an accuracy of 99.97%. The results obtained from the Milimg dataset were 96.7%. The dual dataset approach enabled the model to tap into diverse data sources, enhancing its capability to detect and classify various malware types with greater precision. The achieved accuracy highlights the transformative potential of deep learning in cybersecurity, as these models effectively analyze complex patterns and anomalies within malware, offering a reliable solution against sophisticated threats. These techniques not only enhance detection capabilities but also adapt to the ever-evolving cybersecurity, ensuring a resilient defense against emerging threats. In the future, work will focus on integrating various data types, exploring new deep learning techniques, improving model adaptability to new malware, and enhancing user trust, ultimately making malware detection systems more effective and resilient. The limitations of this research include: a) The EfficientNet model, while effective, is computationally expensive. This may present issues for real-time detection in resource-constrained contexts. b) Processing sensitive data in cybersecurity applications poses privacy problems. Data protection and compliance with legislation are critical. c) The model's capacity to react to new and emerging malware threats over time necessitates regular updates and retraining, which can be resource-intensive. Future research can also focus on exploring new deep learning architectures, incorporating adversarial learning techniques, and addressing the challenges of evolving malware threats.

Acknowledgement: Not applicable.

Funding Statement: This work was supported and funded by Deanship and Scientific Research at Imam Mohammed Ibn Saud Islamic University (IMSIU) (grant number IMSIU-DDRSP2504).

Availability of Data and Materials: The data that support the findings of this study are openly available in [4,5].

Ethics Approval: Not applicable.

Conflicts of Interest: The author declares no conflicts of interest to report regarding the present study.

References

1. Mitchell R, Chen IR. A survey of intrusion detection in wireless network applications. Comput Commun. 2014;42(4):1–23. doi:10.1016/j.comcom.2014.01.012.
2. Rahman MM, Al Shakil S, Mustakim MR. A survey on intrusion detection system in IoT networks. Cyber Secur Appl. 2025;3(1):100082. doi:10.1016/j.csa.2024.100082.
3. Brown A, Gupta M, Abdelsalam M. Automated machine learning for deep learning based malware detection. Comput Secur. 2024;137(1):103582. doi:10.1016/j.cose.2023.103582.
4. Shen L, Feng J, Chen Z, Sun Z, Liang D, Li H, et al. Self-attention based convolutional-LSTM for Android malware detection using network traffics grayscale image. Appl Intell. 2023;53(1):683–705. doi:10.1007/s10489-022-03523-2.
5. Mezina A, Burget R. Obfuscated malware detection using dilated convolutional network. In: 2022 14th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT); 2022 Oct 11–13; Valencia, Spain. p. 110–5. doi:10.1109/ICUMT57764.2022.9943443.
6. Al-Khater W, Al-Madeed S. Using 3D-VGG-16 and 3D-Resnet-18 deep learning models and FABEMD techniques in the detection of malware. Alex Eng J. 2024;89(1):39–52. doi:10.1016/j.aej.2023.12.061.

7. Kosmidis K, Kalloniatis C. Machine learning and images for malware detection and classification. In: Proceedings of the 21st Pan-Hellenic Conference on Informatics; 2017 Sep 28–30; Larissa, Greece. p. 1–6. doi:10.1145/3139367.3139400.
8. Sihag V, Vardhan M, Singh P, Choudhary G, Son S. De-LADY: deep learning based Android malware detection using Dynamic features. J Internet Serv Inf Secur. 2021;11(2):34–45.
9. Alhaidari F, Shaib NA, Alsafi M, Alharbi H, Alawami M, Aljindan R, et al. ZeVigilante: detecting zero-day malware using machine learning and sandboxing analysis techniques. Comput Intell Neurosci. 2022;2022(1):1615528. doi:10.1155/2022/1615528.
10. Jasi TA, Jawhar MM. Detecting network attacks Model based on a long short-term memory LSTM. Technium. 2022;4(8):64–71. doi:10.47577/technium.v4i8.7225.
11. Li T, Luo Y, Wan X, Li Q, Liu Q, Wang R, et al. A malware detection model based on imbalanced heterogeneous graph embeddings. Expert Syst Appl. 2024;246(27):123109. doi:10.1016/j.eswa.2023.123109.
12. Alazab M, Soman KP, Srinivasan S, Venkatraman S, Pham VQ. Deep learning for cyber security applications: a comprehensive survey. arXiv:16748161. 2023.
13. Moutafis I, Andreatos A, Stefaneas P. Spam email detection using machine learning techniques. In: European Conference on Cyber Warfare and Security (ECCWS); 2023 Jun 22–23; Athens, Greece.
14. Syeda DZ, Asghar MN. Dynamic malware classification and API categorisation of windows portable executable files using machine learning. Appl Sci. 2024;14(3):1015. doi:10.3390/app14031015.
15. Karat G, Kannimoola JM, Nair N, Vazhayil A, Sujadevi VG, Poornachandran P. CNN-LSTM hybrid model for enhanced malware analysis and detection. Procedia Comput Sci. 2024;233:492–503. doi:10.1016/j.procs.2024.03.239.
16. Liu Y, Fan H, Zhao J, Zhang J, Yin X. Efficient and generalized image-based CNN algorithm for multi-class malware detection. IEEE Access. 2024;12(5):104317–32. doi:10.1109/ACCESS.2024.3435362.
17. Linh VK, Hùng NV, Anh TN, Nhuan DD, Hien DC. Enhance deep learning model for malware detection with a new image representation method. J Sci Technol Inf Secur. 2024;2024:31–9. doi:10.54654/isj.v1i21.1000.
18. Patil R, Deng W. Malware analysis using machine learning and deep learning techniques. In: 2020 SoutheastCon; 2020 Mar 28–29; Raleigh, NC, USA. p. 1–7. doi:10.1109/southeastcon44009.2020.9368268.
19. Talukder MA, Hasan KF, Islam MM, Uddin MA, Akhter A, Abu Yousuf M, et al. A dependable hybrid machine learning model for network intrusion detection. J Inf Secur Appl. 2023;72(1):103405. doi:10.1016/j.jisa.2022.103405.
20. Thakur P, Kansal V, Rishiwal V. Hybrid deep learning approach based on LSTM and CNN for malware detection. Wirel Pers Commun. 2024;136(3):1879–901. doi:10.1007/s11277-024-11366-y.
21. Singh A, Handa A, Kumar N, Shukla SK. Malware classification using image representation. In: Cyber Security Cryptography and Machine Learning: Third International Symposium, CSCML 2019; 2019 Jun 27–28; Beer-Sheva, Israel. Berlin/Heidelberg, Germany: Springer International Publishing. p. 75–92.
22. Ali J, Khan R, Ahmad N, Maqsood I. Random forests and decision trees. Int J Comput Sci Issues; 2012 Sep 1;9(5):272.
23. Chen T, He T, Benesty M, Khotilovich V, Tang Y, Cho H, et al. Xgboost: extreme gradient boosting. R package version 0.4-2. 2015 Aug 1;1(4):1–4.
24. Tan M, Le Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning; 2019 May 24; PMLR. p. 6105–14.
25. Chollet F. Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2017; PMLR. p. 1251–58.
26. Dener M, Ok G, Orman A. Malware detection using memory analysis data in big data environment. Appl Sci. 2022;12(17):8604. doi:10.3390/app12178604.
27. Ahmad A, Naseer M, Ahmad M, Mehmood K. IMA—an intelligent obfuscated malware analysis model. In: 2024 21st International Bhurban Conference on Applied Sciences and Technology (IBCAST); 2024 Aug 20–23; Murree, Pakistan. p. 98–106. doi:10.1109/IBCAST61650.2024.10877188.

28. Aghaeikheirabady M, Farshchi SMR, Shirazi H. A new approach to malware detection by comparative analysis of data structures in a memory image. In: 2014 International Congress on Technology, Communication and Knowledge (ICTCK); 2014 Nov 26–27; Mashhad, Iran. p. 1–4. doi:10.1109/ICTCK.2014.7033519.
29. Arfeen A, Asim Khan M, Zafar O, Ahsan U. Process based volatile memory forensics for ransomware detection. Concurr Comput Pract Exp. 2022;34(4):e6672. doi:10.1002/cpe.6672.
30. Wang Q, Qian Q. Malicious code classification based on opcode sequences and text CNN network. J Inf Secur Appl. 2022;67(2):103151. doi:10.1016/j.jisa.2022.103151.
31. Taşcı B. Deep-learning-based approach for IoT attack and malware detection. Appl Sci. 2024;14(18):8505. doi:10.3390/app14188505.
32. Sharma A, Babbar H, Vats AK. Securing the Internet of Things: using machine learning for malware detection with CIC-MalMem dataset. In: 2024 4th International Conference on Innovative Practices in Technology and Management (ICIPTM); 2024 Feb 21–23; Noida, India. p. 1–5. doi:10.1109/ICIPTM59628.2024.10563381.
33. Shah SSH, Ahmad AR, Jamil N, Khan AUR. Memory forensics-based malware detection using computer vision and machine learning. Electronics. 2022;11(16):2579. doi:10.3390/electronics11162579.
34. Mashrur Arifin M, Suyehara Tolman T, Yeh JH. Unveiling the efficacy of BERT's attention in memory obfuscated malware detection. In: International Conference on Information Security Practice and Experience; 2024 Oct 25–27; Wuhan, China. Singapore: Springer Nature Singapore; 2024. p. 273–91.
35. Keyes DS, Li B, Kaur G, Lashkari AH, Gagnon F, Massicotte F. EntropLyzer: android malware classification and characterization using entropy analysis of dynamic characteristics. In: 2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS); 2021 May 18–19; Hamilton, ON, Canada. p. 1–12. doi:10.1109/rdaaps48126.2021.9452002.
36. Cao D, Zhang X, Ning Z, Zhao J, Xue F, Yang Y. An efficient malicious code detection system based on convolutional neural networks. In: Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence; 2018 Dec 8–10; Shenzhen, China. p. 86–9. doi:10.1145/3297156.3297246.
37. Awan MJ, Masood OA, Abed Mohammed M, Yasin A, Zain AM, Damaševičius R, et al. Image-based malware classification using VGG19 network and spatial convolutional attention. Electronics. 2021;10(19):2444. doi:10.3390/electronics10192444.