Fırat Üniversitesi Deneysel ve Hesaplamalı Mühendislik Dergisi

Görüntü Tabanlı Kötü Amaçlı Yazılım Sınıflandırması için Hibrit Bir Derin Öğrenme Yaklaşımı

Zülfiye Beyza METİN [1*], Fatih ÖZKAYNAK [2]

[1,2]Yazılım Mühendisliği Bölümü, Teknoloji Fakültesi, Fırat Üniversitesi, Elazığ, Türkiye.
[1]zbmetin@firat.edu.tr, [2]ozkaynak@firat.edu.tr

Geliş Tarihi: 10.11.2024
Kabul Tarihi: 9.08.2025

Düzeltme Tarihi:30.12.2024

doi: https://doi.org/10.62520/fujece.1582676
Araştırma Makalesi

Alıntı: Z. B. Metin ve F. Özkaynak,'' Görüntü tabanlı kötü amaçlı yazılım sınıflandırması için hibrit bir derin öğrenme yaklaşımı", Fırat Üni. Deny. ve Hes. Müh. Derg., vol. 5, no 1, pp. 26-43, Şubat 2026.

Öz

Kötü amaçlı içeriklerin çeşitliliği ve karmaşıklığı, Bilgi ve İletişim Teknolojilerinin son kullanıcılarını önemli ölçüde etkilemiştir. Kötü amaçlı içeriklerin etkisini azaltmak için, kullanıcı sistemlerini kötü amaçlı yazılımlara karşı proaktif olarak korumak üzere otomatikleştirilmiş derin öğrenme tabanlı teknikler geliştirilmiştir. Bu çalışmada, MaleVis veri setini kullanarak kötü amaçlı yazılımların tespiti ve sınıflandırılması için hibrit bir model uyguluyoruz. İlk olarak, veri setinden özellik çıkarma işlemi DenseNet-121, EfficientNet-B0 ve ResNet-50 modelleri kullanılarak gerçekleştirilmiştir. Bu modeller, büyük veri setleri üzerinde eğitilmiş ve güçlü özellik çıkarma yetenekleriyle bilinen derin öğrenme mimarileridir. Her model, Malevis veri setindeki görüntülerden özellik vektörleri çıkarmak için kullanılmıştır. Bu özellik vektörleri daha sonra birleştirilmiştir. Birleştirilen özellik vektörleri, güçlü bir sınıflandırma algoritması olan XGBoost kullanılarak sınıflandırma için kullanılmıştır. Bu hibrit model yaklaşımı, kötü amaçlı yazılımları tespit etmek için derin öğrenme modellerinin özellik çıkarma yeteneklerini XGBoost'un sınıflandırma yeteneği ile birleştirir. Deney sonuçları, önerilen hibrit modelin MaleVis veri setinde yüksek doğruluk oranları elde ettiğini göstermektedir. Çalışma, farklı derin öğrenme modellerinin özellik çıkarma yeteneklerini birleştirmenin ve bu özellikleri XGBoost gibi bir sınıflandırıcıyla kullanmanın kötü amaçlı yazılımların algılanması ve sınıflandırılmasında önemli iyileştirmeler sağlayabileceğini göstermektedir. Sonuçlar, modelin gerçek dünyadaki tehdit algılama sistemlerine entegre edilme potansiyelini ortaya koymaktadır.

Anahtar kelimeler: Kötü amaçlı yazılım, Derin öğrenme, Bilgi ve iletişim teknolojileri, Malevis veri seti

[*]Yazışılan yazar

Firat University Journal of Experimental and Computational Engineering

A Hybrid Deep Learning Approach for Image-based Malware Classification

Zülfiye Beyza METİN [1*], Fatih ÖZKAYNAK [2]

[1,2]Software Engineering, Faculty of Technology, Firat University, Elazığ, Türkiye.
[1]zbmetin@firat.edu.tr, [2]ozkaynak@firat.edu.tr

Received: 10.11.2024
Accepted: 9.08.2025

Revision:30.12.2024

doi: https://doi.org/10.62520/fujece.1582676
Research Article

Citation: Z. B. Metin and F. Özkaynak, "A hybrid deep learning approach for ımage-based malware classification", Firat Univ. Jour.of Exper. and Comp. Eng., vol. 5, no 1, pp. 26-43, February 2026.

Abstract
The diversity and sophistication of malicious content has significantly impacted end-users of Information and Communication Technologies. In order to mitigate the impact of malicious content, automated deep learning-based techniques have been developed to proactively defend user systems against malware. In this study, we implement a hybrid model for malware detection and classification using the MaleVis dataset. First, feature extraction from the dataset is performed using DenseNet-121, EfficientNet-B0 and ResNet-50 models. These models are deep learning architectures that have been trained on large datasets and are known for their powerful feature extraction capabilities. Each model was used to extract feature vectors from the images in the Malevis dataset. These feature vectors were then merged. The combined feature vectors were used for classification using XGBoost, a powerful classification algorithm. This hybrid model approach combines the feature extraction capabilities of deep learning models with the classification capability of XGBoost to detect malware. Experimental results show that the proposed hybrid model achieves high accuracy rates on the MaleVis dataset. The study shows that combining the feature extraction capabilities of different deep learning models and using these features with a classifier such as XGBoost can provide significant improvements in malware detection and classification.The results demonstrate the model's potential for integration into real-world threat detection systems.

Keywords: Malware, Deep learning, Information and communication technologies, Malevis dataset

[*]Corresponging author

1. Introduction

In recent years, the internet has become an integral part of everyday life for almost every member of society. Activities such as social interactions, online banking, health-related transactions, and marketing are almost impossible to carry out without the internet. As the internet has developed rapidly, criminals have increasingly turned to committing crimes online rather than in the physical world [1]. These cybercriminals use malicious software, commonly known as malware, to launch attacks on targeted machines [2]. Any software designed to intentionally execute malicious payloads on devices such as computers, smartphones, and networks is classified as malware [3]. There are several types of malware, including viruses, worms, Trojans, rootkits, and ransomware, each designed to infect the compromised system in different ways, such as damaging the system, enabling remote code execution, or stealing confidential data [4].

The classification of malware has become increasingly difficult due to the emergence of malware that exhibits characteristics of more than one type simultaneously. In the early days, malware was created for simpler purposes and was therefore easier to detect, often referred to as traditional or simple malware. Today, however, malware has evolved into more destructive and harder-to-detect forms, known as next-generation malware [5]. This new generation of malware can run in kernel mode and bypass protection software such as firewalls and antivirus programs. Unlike traditional malware, which usually consists of a single process and uses simple techniques, next-generation malware employs multiple processes and obfuscation techniques to hide and persist in the system [6]. These advanced malware strains can launch unprecedented and more devastating attacks, often involving multiple malware strains simultaneously [7].

Detecting these malicious programs is essential to protect users and organizations from malware [8]. Malware detection is the process of determining whether a particular program is malicious. Initially, signature-based detection approaches were widely used, but they have limitations, especially in detecting unknown and next-generation malware. Over time, researchers have developed new approaches such as behavioral, heuristic, and model-checking based detection. Data mining and machine learning (ML) algorithms are also increasingly applied in malware detection [9]. Recently, advanced techniques based on deep learning have emerged [10], showing promise in detecting both known and unknown malware [11]. Several studies have explored image-based malware classification. Nataraj et al. [12] visualized malware binaries as grayscale images and used similarity calculations to classify malware into families, achieving 98% classification accuracy. Similarly, Makandar et al. [13] applied Support Vector Machines (SVM) for malware classification, reaching 98.88% accuracy. Ni et al. [14] converted malware opcodes into grayscale images and achieved 99.26% accuracy. Qiao et al. [15] visualized malware binaries as multi-channel images and employed the LeNet5 model, obtaining 98.76% accuracy. Xiao et al. achieved 99.7% accuracy using deep learning-based feature extraction methods. Vu et al. [16] proposed pixel encoding and byte arrangement from binaries into images using space-filling curves, combining entropy coding and a character class scheme over the Hilbert curve, and achieved 93.01% accuracy. Vasan et al. [17] transformed malware binaries into colored images for multi-class classification using an ESA architecture, transfer learning, and data augmentation, achieving 98.82% accuracy on the Malimg dataset and 97.35% on an IoT Android dataset, noting that colored images outperformed grayscale. Khan et al. [18] classified malware on MMCC datasets using ESA architectures (Inceptionv4, ResNet18, ResNet34, ResNet50, ResNet101, ResNet152) and found that ResNet152 achieved the best test accuracy of 88.36%.

Machine learning and deep learning-based methods have proven effective in detecting malware, especially in large datasets [19]. However, each method has limitations. Therefore, hybrid approaches that combine different methods can enhance detection. Hybrid models leverage the feature extraction capabilities of deep learning models with robust classifiers such as XGBoost, providing higher accuracy in malware detection. In this study, we propose a hybrid model for malware detection using pre-trained deep learning models, including ResNet-50, EfficientNet-B0, and DenseNet-121. Features extracted by these models are combined with an XGBoost classifier to classify malware. The aim of the study is to enhance existing detection methods and achieve higher accuracy in malware detection.

2. Material and Methods

This paper proposes a deep learning-based hybrid method for the automatic and effective detection of malware. In the proposed method, we process input data using state-of-the-art and popular deep learning models such as ResNet-50, EfficientNet-B0, and DenseNet-121. These models are selected primarily for their image-based and generalizable feature extraction capabilities, which demonstrate high performance across different datasets. Each network extracts meaningful and abstract features from the input data and represents them as high-dimensional feature vectors. However, a summarization step is required to make these features directly usable for classification. For this purpose, we employ a Global Average Pooling layer to summarize feature maps.

We then combine the summarized feature vectors during the feature fusion stage, allowing the strengths of each deep learning model to be utilized, resulting in a rich, unified feature vector. This step enables the capture of more meaningful and discriminative features by consolidating the inference power of different networks under a single representation. As a result, the classification performance is significantly improved. In the final stage, we train the combined feature vector using XGBoost, a powerful gradient boosting algorithm. XGBoost demonstrates superior performance compared to other traditional machine learning algorithms due to its high discriminative capacity and ability to model complex data relationships. This computationally efficient algorithm effectively handles noise in the dataset, enabling more accurate classifications.A block diagram of the proposed hybrid method is presented in Figure 1.

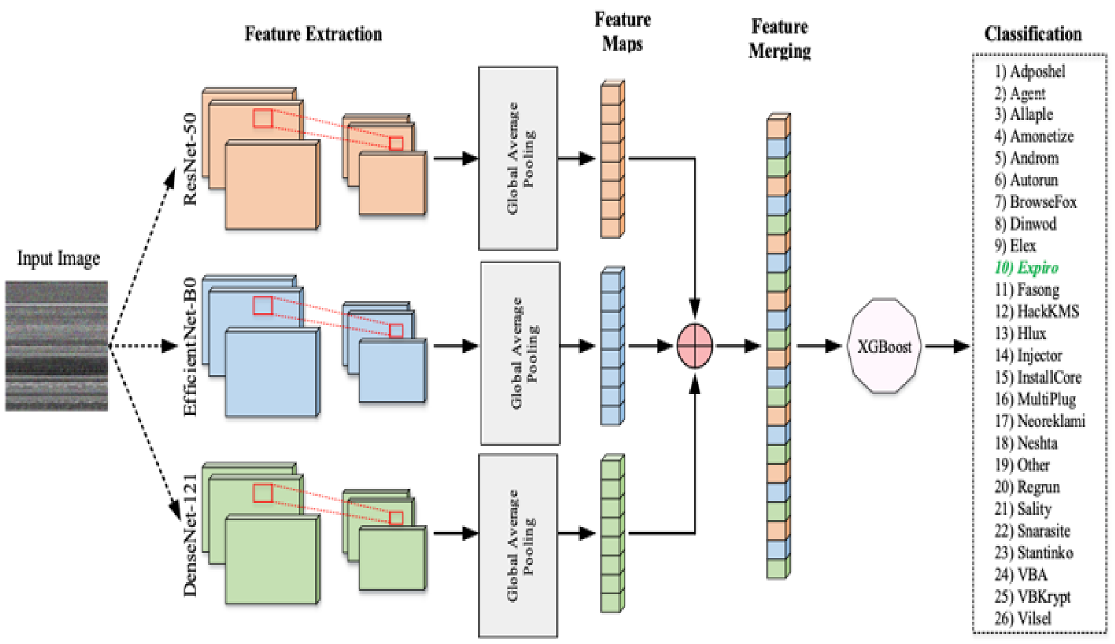

Figure 1. Proposed hybrid method

2.1. Dataset

In this study uses the Malware Evaluation with Vision (Malevis) dataset. The Malevis dataset is a pioneering resource developed for researchers focusing on the detection of multi-class malware through image-based approaches. It provides an RGB-based real data set specifically designed to facilitate the evaluation of various malware detection efforts. The Malevis dataset contains images from 26 different classes, where one class represents “legitimate” samples and the remaining 25 classes correspond to various types of malware. The dataset was converted to RGB images using the Bin2png library as described in [20]. The steps taken to create the Malevis dataset are summarized in Figure 2.

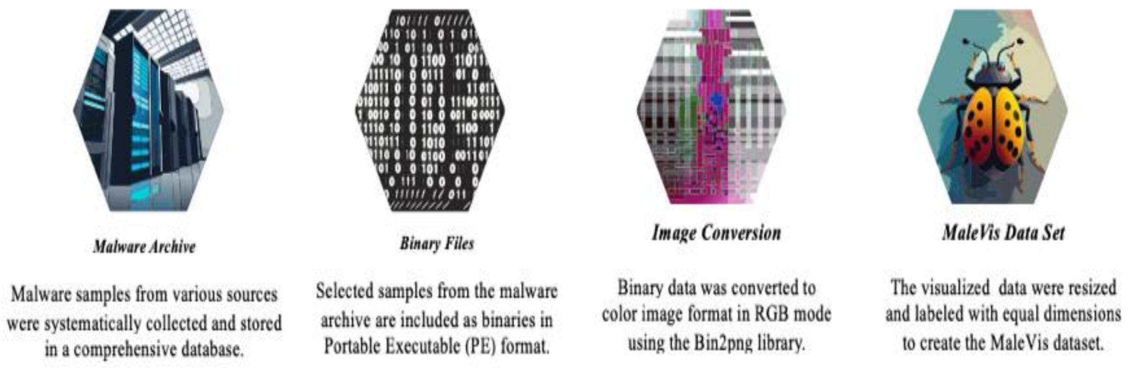
Malware Archive
Malware samples from various sources were systematically collected and stored in a comprehensive database.
Binary Files
Selected samples from the malware archive are included as binaries in Portable Executable (PE) format.
Image Conversion
Binary data was converted to color image format in RGB mode using the Bin2png library.
MaleVis Data Set
The visualized data were resized and labeled with equal dimensions to create the MaleVis dataset.

Figure 2. Steps in creating the malevis dataset

The Malevis dataset has a balanced distribution with 500 or close to 500 samples for each class. Randomly selected examples among the Malevis dataset classes are presented in Figure 3.

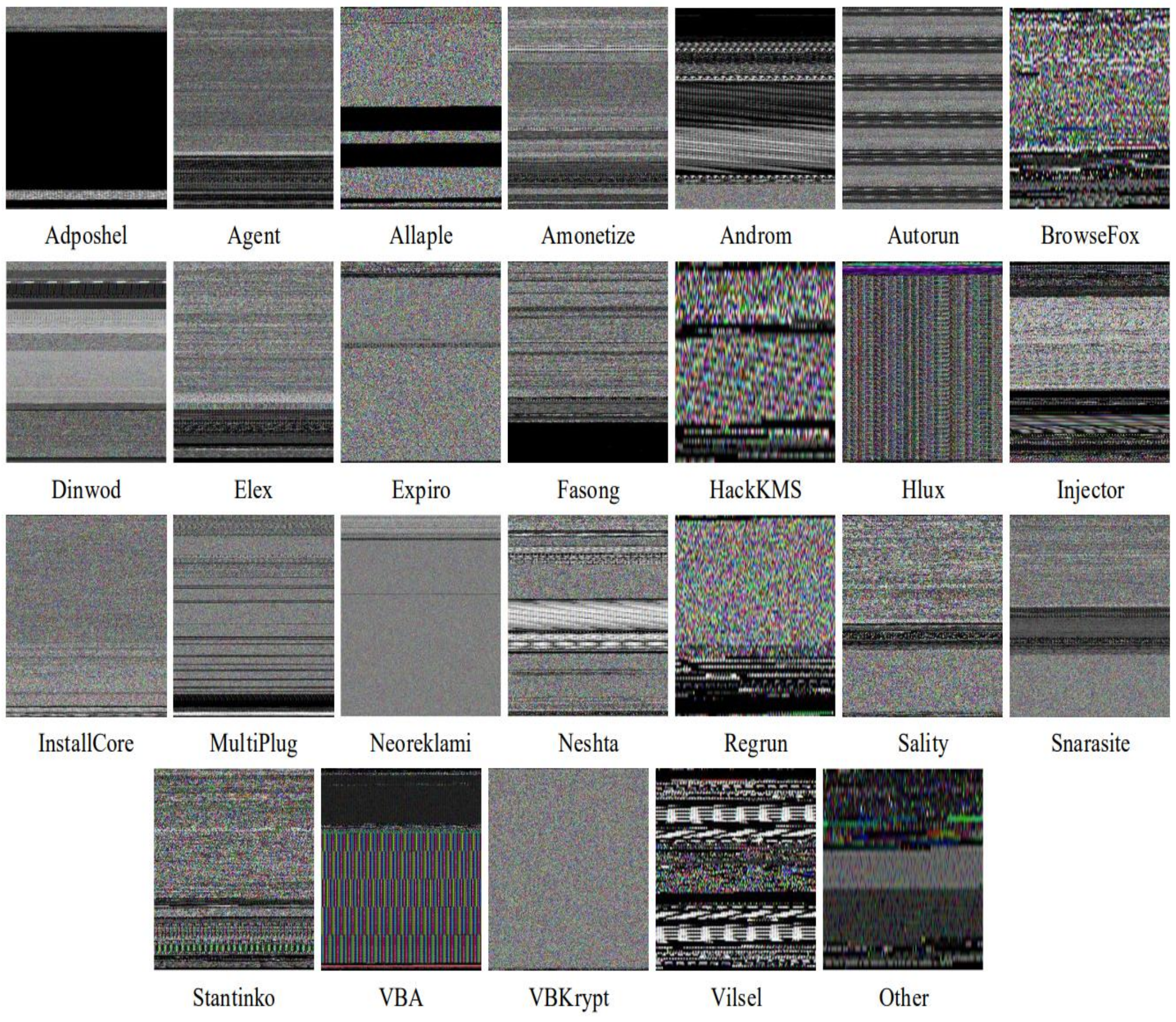

Figure 3. Malevis dataset samples

Table 1. Distribution of data set samples

Category	Class Name	Number of Samples
Adware	Adposhel	494
	Amonetize	497
	BrowseFox	493
	InstallCore	500
	MultiPlug	499
	Neoreklami	500
Trojan	Agent	470
	Dinwod	499
	Elex	500
	HackKMS	499
	Injector	495
	Regrun	485
	Snarasite	500
	VBKrypt	496
	Vilsel	496
Worm	Allaple	478
	Autorun	496
	Fasong	500
	Hlux	500
Backdoor	Androm	500
	Stantinko	500
Virus	Expiro	501
	Neshta	497
	Sality	499
	VBA	500
Normal	Other	1832

The dataset contains a total of 9100 training images and 5126 validation images. Each training class contains 350 image samples, while the validation set contains various numbers of images per class. The class-based distribution of the Malevis dataset samples is given in Table 1.

2.2. Pretrained models

Pretrained models have become a major focus of interest in the fields of artificial intelligence and machine learning in recent years. These models are systems that are pre-trained on large datasets and then adapted for a specific task. This process is called transfer learning and is widely used, especially in deep learning applications. The main advantage of pre-trained models is that they eliminate the need to restart the training process, which requires large amounts of data and high computational power. These models are usually trained on large and diverse datasets, which allows them to have a large knowledge base. Furthermore, pre-trained models allow for better performance with less data. This is a big advantage, especially in areas where data collection is difficult or costly. Thanks to transfer learning, a pre-trained model can be retrained on a small dataset to achieve high performance on a given task [21]. Popular representatives of pre-trained models include architectures such as ResNet, EfficientNet and DenseNet [22].

2.2.1. ResNet architecture

In traditional neural network architectures, the output of each layer is directly used as input for the next layer. In this approach, as the network depth increases, problems such as vanishing gradients may occur. To address these issues, the ResNet architecture was developed in 2015 by Kaiming and his colleagues. In the ResNet architecture, residual blocks are formed by connecting two layers. The second layer directly receives the output of the first layer and adds it to the input from the initial layer. This allows errors occurring at any layer of the network to propagate back to the input layer, thereby preventing issues such as vanishing gradients [23].

2.2.2. EfficientNet architecture

ESAs are typically developed with an initial fixed resource cost, and as more resources become available, the number of layers is increased to achieve better accuracy. In addition to increasing the number of layers, enhancing the model's depth or width and using higher input resolution for training and evaluation can improve accuracy. However, these methods may still be insufficient in terms of performance, even if they enhance accuracy. Tan and colleagues proposed a novel model scaling method that uses a compound coefficient to systematically scale ESAs. Instead of independently increasing depth, width, or input resolution, this method adjusts these dimensions using a fixed scaling factor [24]. With this new scaling method, the EfficientNet model family, which is smaller and faster compared to other models, was developed. The EfficientNet family consists of eight models named sequentially from EfficientNet-B0 to EfficientNet-B7. EfficientNet-B0 was designed as the base model, and all other models are scaled versions of it [25].

2.2.3. DenseNet architecture

The DenseNet architecture, proposed by Huang and colleagues, features an innovative structure where layers are densely connected. In this architecture, each layer receives the outputs of all preceding layers as input. This densely connected structure facilitates better gradient propagation and enhances feature reusability, thereby significantly improving the model's accuracy and efficiency. The DenseNet architecture has been further optimized through various scaling methods. Adjusting dimensions such as depth, width, and input resolution using a fixed scaling factor improves the model's performance and accuracy [26]. Different variants of DenseNet are sequentially named from DenseNet-121 to DenseNet-201, each representing a scaled version of the base model. This scaling strategy ensures that DenseNet models are smaller and more efficient, offering superiority in terms of both accuracy and performance [27].

2.3. Evaluation metrics

A frequently used method to evaluate the performance of deep learning models on classification tasks is the confusion matrix. The confusion matrix reveals the correspondence between the class label predicted by the model for a given input image and the actual class label of that image. In AI-based classification studies, various results can be obtained at the output layer of the model. These results can be summarized in four basic metrics: The areas represented by the relevant metrics of a randomly selected class used in this study are shown in Figure 4 on the confusion matrix.

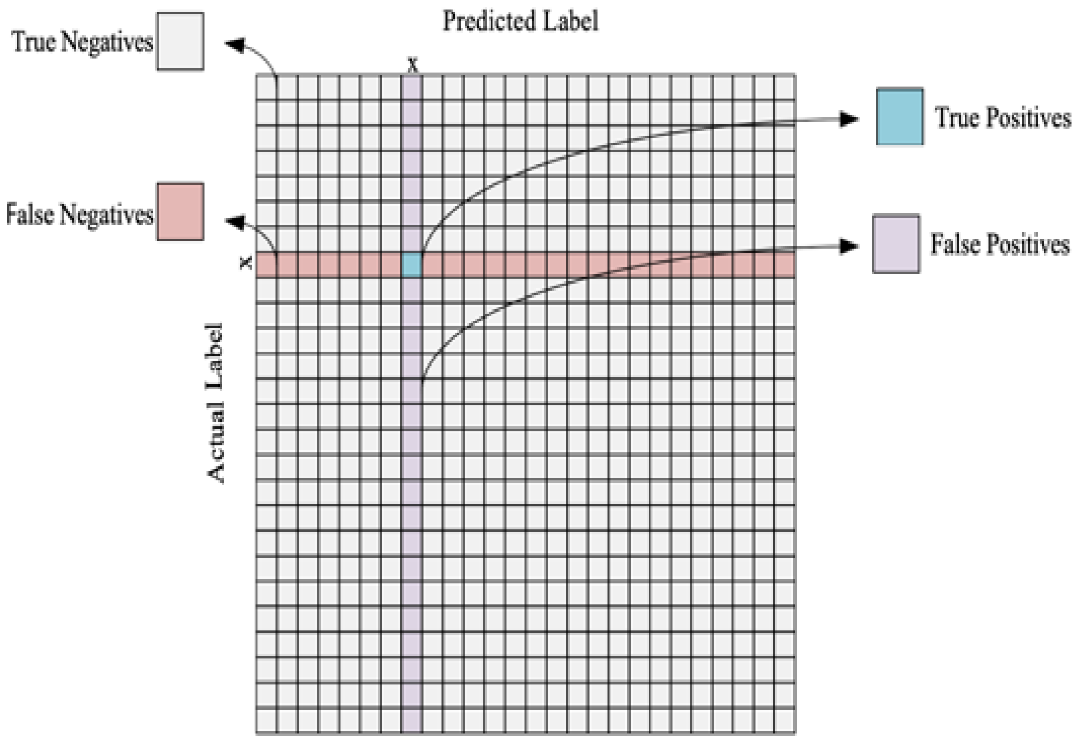

Figure 4. Confusion matrix

Various metrics such as accuracy, precision, sensitivity and F-1 score can be used to evaluate the performance of an image classification deep learning model. Accuracy refers to the rate at which the model makes correct predictions and is calculated as the ratio of correct predictions to total predictions. It is calculated using Equation 1. Recall measures the capacity of the model to accurately detect all positive cases and is calculated as the ratio of true positive predictions to total true positive cases. It is calculated using Equation 2. Precision measures the proportion of cases that the model correctly identifies as positive and is calculated as the ratio of true positive predictions to total true positive predictions. It is calculated using Equation 3. The F-1 score, the harmonic mean of sensitivity and precision, balances these two metrics and provides a useful measure in cases of overlap. It is calculated using Equation 4. The mathematical formulas for these performance metrics are presented below.

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

$$\text{Recall} = \left(\frac{\text{TP}}{\text{TP+FN}}\right) \tag{2}$$

$$\text{Precision} = \left(\frac{\text{TP}}{\text{TP+FP}}\right) \tag{3}$$

$$\text{F1 Score} = \left(\frac{2 \times \text{Pre} \times \text{Rec}}{\text{Pre+Rec}}\right) \tag{4}$$

3. Experiments

The experiments carried out to evaluate the performance of the hybrid model are detailed in this section. Furthermore, the following sections present the analysis of the experimental results and the evaluation of the performance metrics.

3.1. Experimental setup

The dataset samples can be randomly separated using 70% training, 20% working and 10% testing methods. Then, pre-trained ResNet-50, EfficientNet-B0 and DenseNet-121 models with ImageNet weights were included in the Python environment. It was shown that each model was classified and validated with its default layers and hyperparameters for a maximum of 100 steps. However, the properties of the early termination functions and the accuracy value obtained by combining them were monitored. If there is no decrease in accuracy during the five consecutive steps, the early termination function is activated and the model's training is terminated. In this way, the weights of the step with the lowest accuracy values exposed to excessive explosions are recorded. The detailed layout for this study is given in Figure 5. The hyperparameters for the XGBoost based classifier were determined through a systematic evaluation process in order to increase the generalization capability of the model and prevent overlearning; different parameter combinations were compared through multiple trials and the configuration that provided the highest accuracy performance was preferred.

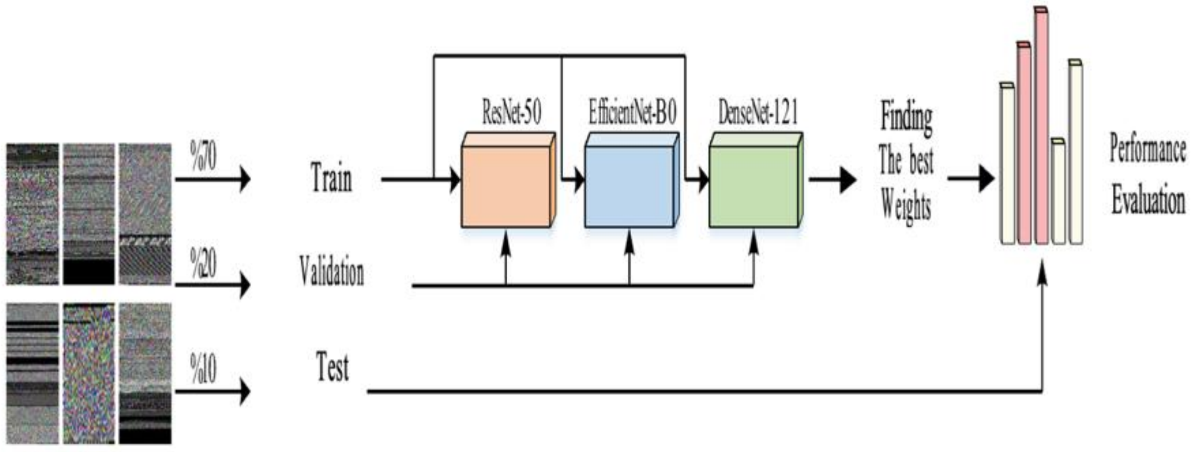

Figure 5. Experimental layout steps

3.2. Results

ResNet-50, EfficientNet-B0 and DenseNet-121 deep learning models were trained for a maximum of 100 rounds with the help of an early termination function. The weights at the end of the round are recorded andthe resulting loss and accuracy graphs are given in Figure 6. The loss and accuracy graphs in Figure 6 clearly show how the ResNet-50, EfficientNet-B0 and DenseNet-121 models performed during the training period and in which rounds their performance improved or stagnated. Through these graphs, critical points such as which model learns faster, which one achieves higher accuracy rates, and how their losses change during training can be analyzed.

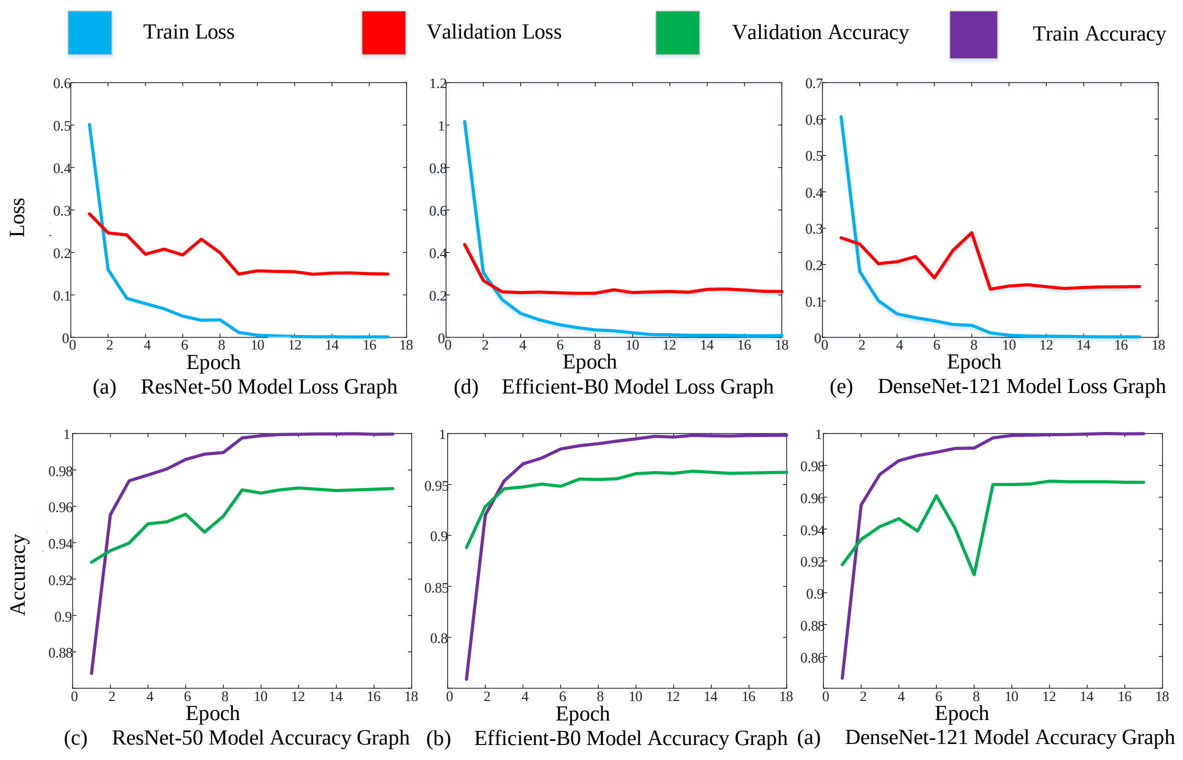

Figure 6. Loss and accuracy graph of ResNet-50, DenseNet-121 and EfficientNet-B0

Only the images allocated for the test phase were given as input to the ResNet-50 model. The confusion matrix created with reference to the model's predictions for these test images is given in Figure 7.

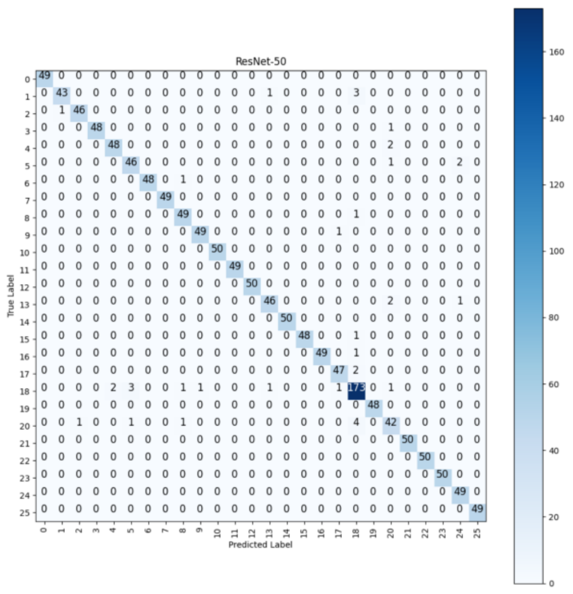

Figure 7. Confusion matrix of ResNet-50

Table 2 shows the values of the performance metrics achieved by the ResNet-50 model during testing. The ResNet-50 model achieved the highest verification accuracy rate of 97.32%.

Table 2. Performance values of the ResNet-50

Class (Label)	Precision	Recall	F-1 Score
Adposhel (0)	1	1	1
Agent (1)	0.977273	0.914894	0.945055
Allaple (2)	0.978723	0.978723	0.978723
Amonetize (3)	1	0.979592	0.989691
Androm (4)	0.96	0.96	0.96
Autorun (5)	0.92	0.938776	0.929293
BrowseFox (6)	1	0.979592	0.989691
Dinwod (7)	1	1	1
Elex (8)	0.942308	0.98	0.960784
Expiro (9)	0.98	0.98	0.98
Fasong (10)	1	1	1
HackKMS (11)	1	1	1
Hlux (12)	1	1	1
Injector (13)	0.958333	0.938776	0.948454
InstallCore (14)	1	1	1
MultiPlug (15)	1	0.979592	0.989691
Neoreklami (16)	1	0.98	0.989899
Neshta (17)	0.959184	0.959184	0.959184
Other (18)	0.935135	0.945355	0.940217
Regrun (19)	1	1	1
Sality (20)	0.857143	0.857143	0.857143
Snarasite (21)	1	1	1
Stantinko (22)	1	1	1
VBA (23)	1	1	1
VBKrypt (24)	0.942308	1	0.970297
Vilsel (25)	1	1	1

Only the images allocated for the test phase are given as input to the EfficientNet-B0 model. The confusion matrix created with reference to the model's predictions for these test images is given in Figure 8.

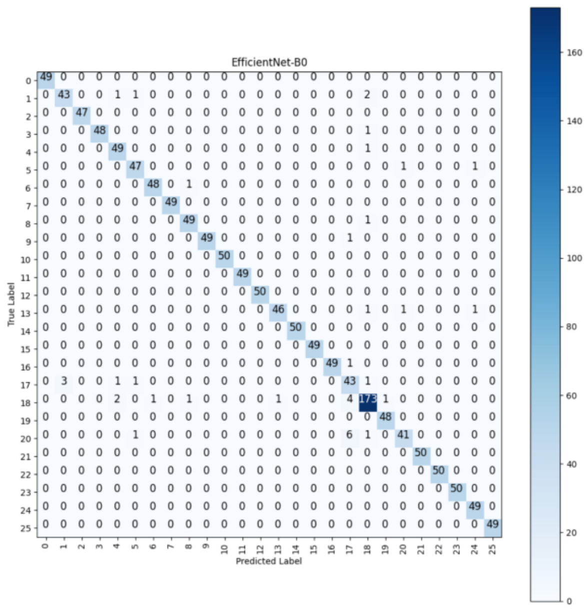

Figure 8. Confusion matrix of EfficientNet-B0

Table 3 shows the values of the performance metrics achieved by the EfficientNet-B0 model during testing. The EfficientNet-B0 model achieved the highest verification accuracy rate with 97.24%.

Table 3. Performance values of the EfficientNet-B0

Class (Label)	Precision	Recall	F-1 Score
Adposhel (0)	1	1	1
Agent (1)	0.934783	0.914894	0.924731
Allaple (2)	1	1	1
Amonetize (3)	1	0.979592	0.989691
Androm (4)	0.924528	0.98	0.951456
Autorun (5)	0.94	0.959184	0.949495
BrowseFox (6)	0.979592	0.979592	0.979592
Dinwod (7)	1	1	1
Elex (8)	0.960784	0.98	0.970297
Expiro (9)	1	0.98	0.989899
Fasong (10)	1	1	1
HackKMS (11)	1	1	1
Hlux (12)	1	1	1
Injector (13)	0.978723	0.938776	0.958333
InstallCore (14)	1	1	1
MultiPlug (15)	1	1	1
Neoreklami (16)	1	0.98	0.989899
Neshta (17)	0.781818	0.877551	0.826923

Table 3. (Contineu) Performance values of the EfficientNet-B0

Class (Label)	Precision	Recall	F-1 Score
Other (18)	0.955801	0.945355	0.950549
Regrun (19)	0.979592	1	0.989691
Sality (20)	0.953488	0.836735	0.891304
Snarasite (21)	1	1	1
Stantinko (22)	1	1	1
VBA (23)	1	1	1
VBKrypt (24)	0.960784	1	0.98
Vilsel (25)	1	1	1

Only the images allocated for the test phase were given as input to the DenseNet-121 model. The confusion matrix created with reference to the model's predictions for these test images is given in Figure 9.

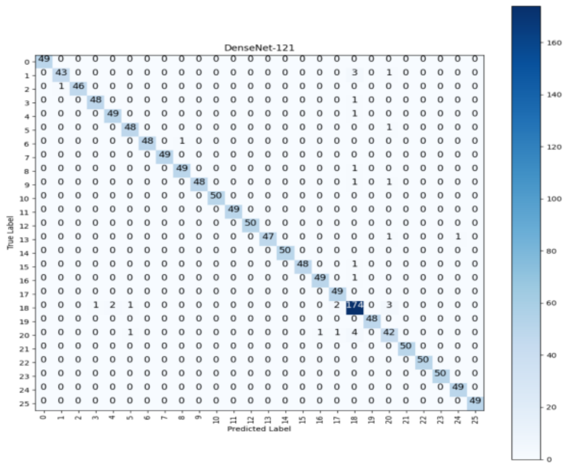

Figure 9. Confusion matrix of DenseNet-121

Table 4 shows the values of the performance metrics achieved by the DenseNet-121 model during testing. The DenseNet-121 model achieved the highest verification accuracy rate with 97.74%.

Table 4. Performance values of the DenseNet-121

Class (Label)	Precision	Recall	F-1 Score
Adposhel (0)	1	1	1
Agent (1)	0.977273	0.914894	0.945055
Allaple (2)	1	0.978723	0.989247
Amonetize (3)	0.979592	0.979592	0.979592
Androm (4)	0.960784	0.98	0.970297
Autorun (5)	0.96	0.979592	0.969697
BrowseFox (6)	1	0.979592	0.989691
Dinwod (7)	1	1	1

Table 4. (Continue) Performance values of the DenseNet-121

Class (Label)	Precision	Recall	F-1 Score
Elex (8)	0.98	0.98	0.98
Expiro (9)	1	0.96	0.979592
Fasong (10)	1	1	1
HackKMS (11)	1	1	1
Hlux (12)	1	1	1
Injector (13)	1	0.959184	0.979167
InstallCore (14)	1	1	1
MultiPlug (15)	1	0.979592	0.989691
Neoreklami (16)	0.98	0.98	0.98
Neshta (17)	0.942308	1	0.970297
Other (18)	0.930481	0.95082	0.940541
Regrun (19)	1	1	1
Sality (20)	0.857143	0.857143	0.857143
Snarasite (21)	1	1	1
Stantinko (22)	1	1	1
VBA (23)	1	1	1
VBKrypt (24)	0.98	1	0.989899
Vilsel (25)	1	1	1

Only the images allocated for the test phase are given as input to the proposed hybrid model. The confusion matrix based on the model's predictions for these test images is given in Figure 10.

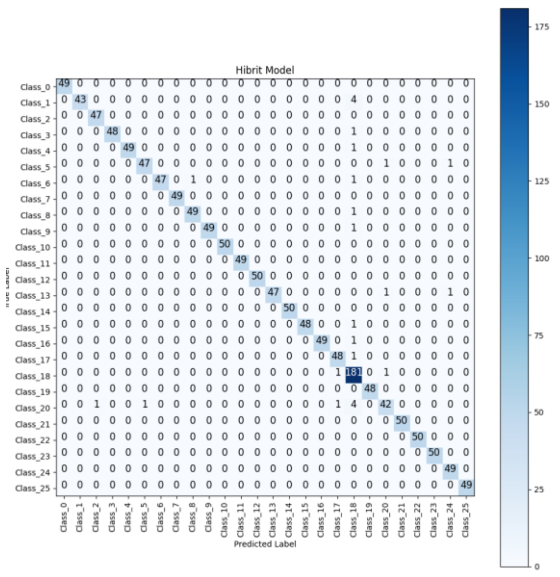

Figure 10. Confusion matrix of the proposed hybrid model

Table 5. Performance values of the proposed hybrid model

Class (Label)	Precision	Recall	F-1 Score
Adposhel (0)	1	1	1
Agent (1)	1	0.914894	0.955556
Allaple (2)	0.979167	1	0.989474
Amonetize (3)	1	0.979592	0.989691
Androm (4)	1	0.98	0.989899
Autorun (5)	0.979167	0.959184	0.969072
BrowseFox (6)	1	0.959184	0.979167
Dinwod (7)	1	1	1
Elex (8)	0.98	0.98	0.98
Expiro (9)	1	0.98	0.989899
Fasong (10)	1	1	1
HackKMS (11)	1	1	1
Hlux (12)	1	1	1
Injector (13)	1	0.959184	0.979167
InstallCore (14)	1	1	1
MultiPlug (15)	1	0.979592	0.989691
Neoreklami (16)	1	0.98	0.989899
Neshta (17)	0.96	0.979592	0.969697
Other (18)	0.918782	0.989071	0.952632
Regrun (19)	1	1	1
Sality (20)	0.933333	0.857143	0.893617
Snarasite (21)	1	1	1
Stantinko (22)	1	1	1
VBA (23)	1	1	1
VBKrypt (24)	0.960784	1	0.98
Vilsel (25)	1	1	1

Table 5 shows the values of the performance metrics achieved by the proposed hybrid model during testing. The proposed hybrid model achieved the highest verification accuracy rate of 98.16%.

4. Discussion

In recent years, malware classification has become a critical concern in the field of cybersecurity. Traditional detection methods often fall short in addressing the rapidly evolving and increasingly sophisticated nature of malware. Consequently, the adoption of machine learning and deep learning techniques has gained significant momentum, as these methods offer enhanced accuracy and adaptability in threat detection.

This study introduces a hybrid deep learning approach for malware classification. The proposed model integrates feature extraction from three pre-trained CNN architectures DenseNet-121, EfficientNet-B0, and ResNet-50 and employs XGBoost as the final classifier. This combination enables the extraction of complementary and high-level feature representations, while leveraging the discriminative capability of XGBoost for robust decision making. The effectiveness of the proposed approach is evidenced by a test accuracy of 98.16%, which demonstrates a performance gain over earlier studies. For instance, Tuncer et al. [28] proposed a byte-code-based malware recognition method utilizing Local Neighborhood Binary Pattern (LNBP), achieving 89.40% accuracy. In another study, they employed Local Binary Pattern (LBP), Singular Value Decomposition (SVD), and a Local Ternary Pattern Network (LTPNet), reaching 88.08% accuracy [29]. Compared to these prior approaches, the hybrid model proposed in this work achieves higher classification accuracy and improved robustness, making it more suitable for practical cybersecurity applications. However, potential limitations should be noted: the MaleVis dataset may not fully capture the diversity of real-world malware, which could introduce dataset bias; scalability to larger or streaming data environments may require additional optimization; and real-world deployment would necessitate considerations of latency and resource constraints.

5. Conclusion

In this study, a deep learning-based hybrid model for automatic malware detection is proposed. The proposed model is applied on the MaleVis dataset. In the proposed method, we extract features from input images using ResNet-50, EfficientNet-B0, and DenseNet-121 models, and these features are summarized by global mean pooling. The resulting feature maps are combined in the feature fusion stage to create a richer feature vector. Finally, this fused feature vector is trained and classified using the XGBoost algorithm to accurately identify different types of malware. As a result of the experimental evaluation, the proposed hybrid model achieved a test accuracy of 98.16%. The results demonstrate that integrating the features obtained by deep learning models with a powerful classifier such as XGBoost is an effective strategy for detecting next-generation threats in cybersecurity. Moreover, the findings indicate that the proposed model shows strong potential for deployment in real-time malware detection frameworks, while also being adaptable for larger datasets and broader application domains in future research.

6. Acknowledgements

This paper is supported by Fırat University Scientific Research Projects Coordination Unit (FUBAP) with project number TEKF 24.27 and Scientific and Technological Research Council of Turkey (TUBITAK) with project number 122E337. Also, the authors sincerely thank the editor(s) and the anonymous reviewers contributed positively to this paper's academic development with their constructive and quick feedback.

7. Author Contribution Statement

Author 1 contributed to the preparation of the original draft of the study, methodology development, visualization and experimentation stages. Author 2 conceptualized the study and participated in the writing, revising and editing processes.

8. Ethics Committee Approval and Conflict of Interest

Ethics committee permission is not required for the prepared article. There is no conflict of interest with any person/institution in the prepared article.

9. Ethical Statement Regarding the Use of Artificial Intelligence

No artificial intelligence-based tools or applications were used in the preparation of this study. The entire content of the study was produced by the authors in accordance with scientific research methods and academic ethical principles.

10. References

[1] A. Di Nicola, “Towards digital organized crime and digital sociology of organized crime,” Trends Organ. Crime, pp. 1–20, 2022.
[2] Y. Zhang, Y. Xiao, K. Ghaboosi, J. Zhang, and H. Deng, “A survey of cyber crimes,” Secur. Commun. Netw., vol. 5, no. 4, pp. 422–437, 2012.
[3] A. Chakraborty, A. Biswas, and A. K. Khan, “Artificial intelligence for cybersecurity: Threats, attacks and mitigation,” in Artificial Intelligence for Societal Issues. Cham, Switzerland: Springer, 2023, pp. 3–25.
[4] Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, “A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions,” Electron., vol. 12, no. 6, p. 1333, 2023.
[5] E. M. Rudd, A. Rozsa, M. Günther, and T. E. Boult, “A survey of stealth malware attacks, mitigation measures, and steps toward autonomous open world solutions,” IEEE Commun. Surv. Tutor., vol. 19, no. 2, pp. 1145–1172, 2016.
[6] S. S. Chakkaravarthy, D. Sangeetha, and V. Vaidehi, “A survey on malware analysis and mitigation techniques,” Comput. Sci. Rev., vol. 32, pp. 1–23, 2019.
[7] J. G. Heiser, “Understanding today’s malware,” Inf. Secur. Tech. Rep., vol. 9, no. 2, pp. 47–64, 2004.
[8] F. A. Aboaoja, A. Zainal, F. A. Ghaleb, B. A. S. Al-Rimy, T. A. E. Eisa, and A. A. H. Elnour, “Malware detection issues, challenges, and future directions: A survey,” Appl. Sci., vol. 12, no. 17, p. 8482, 2022.
[9] M. I. Malik, A. Ibrahim, P. Hannay, and L. F. Sikos, “Developing resilient cyber-physical systems: A review of state-of-the-art malware detection approaches, gaps, and future directions,” Computers, vol. 12, no. 4, p. 79, 2023.
[10] A. Thakkar and R. Lohiya, “A review on machine learning and deep learning perspectives of IDS for IoT: Recent updates, security issues, and challenges,” Arch. Comput. Methods Eng., vol. 28, no. 4, pp. 3211–3243, 2021.
[11] Z. Chen et al., “Machine learning-enabled IoT security: Open issues and challenges under advanced persistent threats,” ACM Comput. Surv., vol. 55, no. 5, pp. 1–37, 2022.
[12] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, “Malware images: Visualization and automatic classification,” in Proc. 8th Int. Symp. Visualization Cyber Secur., 2011, pp. 1–7.
[13] M. Goyal and R. Kumar, “IVMCT: Image visualization based multiclass malware classification using transfer learning,” Math. Stat. Eng. Appl., vol. 71, no. 2, pp. 42–50, 2022.
[14] S. Jang, S. Li, and Y. Sung, “FastText-based local feature visualization algorithm for merged image-based malware classification framework for cyber security and cyber defense,” Mathematics, vol. 8, no. 3, p. 460, 2020.
[15] Y. Qiao, Q. Jiang, Z. Jiang, and L. Gu, “A multi-channel visualization method for malware classification based on deep learning,” in Proc. IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom), 2019, pp. 757–762.
[16] D. L. Vu, T. K. Nguyen, T. V. Nguyen, T. N. Nguyen, F. Massacci, and P. H. Phung, “HIT4Mal: Hybrid image transformation for malware classification,” Trans. Emerg. Telecommun. Technol., vol. 31, no. 11, p. e3789, 2020.
[17] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, and Q. Zheng, “IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture,” Comput. Netw., vol. 171, p. 107138, 2020.
[18] R. U. Khan, X. Zhang, and R. Kumar, “Analysis of ResNet and GoogLeNet models for malware detection,” J. Comput. Virol. Hack. Tech., vol. 15, no. 1, pp. 29–37, 2019.
[19] U.-H. Tayyab, F. B. Khan, M. H. Durad, A. Khan, and Y. S. Lee, “A survey of the recent trends in deep learning based malware detection,” J. Cybersecur. Priv., vol. 2, no. 4, pp. 800–829, 2022.
[20] O. Katar and Ö. Yıldırım, “Classification of malware images using fine-tuned ViT,” Sakarya Univ. J. Comput. Inf. Sci., vol. 7, no. 1, pp. 22–35, 2024.
[21] A. Hosna, E. Merry, J. Gyalmo, Z. Alom, Z. Aung, and M. A. Azim, “Transfer learning: A friendly introduction,” J. Big Data, vol. 9, no. 1, p. 102, 2022.

[22] A. Ait Nasser and M. A. Akhloufi, “A review of recent advances in deep learning models for chest disease detection using radiography,” Diagnostics, vol. 13, no. 1, p. 159, 2023.
[23] B. Koonce, “ResNet-50,” in Convolutional Neural Networks with Swift for TensorFlow: Image Recognition and Dataset Categorization. Berkeley, CA, USA: Apress, 2021, pp. 63–72.
[24] B. Koonce, “EfficientNet,” in Convolutional Neural Networks with Swift for TensorFlow: Image Recognition and Dataset Categorization. Berkeley, CA, USA: Apress, 2021, pp. 109–123.
[25] H. Garg, B. Sharma, S. Shekhar, and R. Agarwal, “Spoofing detection system for e-health digital twin using EfficientNet convolutional neural network,” Multimedia Tools Appl., vol. 81, no. 19, pp. 26873–26888, 2022.
[26] J. Hemalatha, S. A. Roseline, S. Geetha, S. Kadry, and R. Damaševičius, “An efficient DenseNet-based deep learning model for malware detection,” Entropy, vol. 23, no. 3, p. 344, 2021.
[27] H. A. Sanghvi, R. H. Patel, A. Agarwal, S. Gupta, V. Sawhney, and A. S. Pandya, “A deep learning approach for classification of COVID and pneumonia using DenseNet-201,” Int. J. Imaging Syst. Technol., vol. 33, no. 1, pp. 18–38, 2023.
[28] T. Tuncer, F. Ertam, and S. Doğan, “Automated malware recognition method based on local neighborhood binary pattern,” Multimedia Tools Appl., vol. 79, no. 37, pp. 27815–27832, 2020.
[29] T. Tuncer, F. Ertam, and S. Doğan, “Automated malware identification method using image descriptors and singular value decomposition,” Multimedia Tools Appl., vol. 80, no. 7, pp. 10881–10900, 2021.