SAKARYA UNIVERSITY JOURNAL OF COMPUTER AND INFORMATION SCIENCES

http://saucis.sakarya.edu.tr/

e-ISSN: 2636-8129
Publisher: Sakarya University

Vol. 7, No. 1, 2024, 22-35
DOI: 10.35377/saucis...1341082

RESEARCH ARTICLE

Classification of Malware Images Using Fine-Tunned ViT

Oğuzhan Katar [1], Özal Yıldırım [2]

[1] Department of Software Engineering, Faculty of Technology, Fırat University, Elazığ, Türkiye
[2] Department of Artificial Intelligence and Data Engineering, Faculty of Engineering, Fırat University, Elazığ, Türkiye

Corresponding author:
Oguzhan Katar, Department of Software Engineering, Faculty of Technology, Firat University, Elazig, Türkiye
okatar@firat.edu.tr

Article History:
Received: 10.08.2023
Accepted: 25.01.2024
Published Online: 25.01.2024

ABSTRACT
Identifying and classifying malware has become a critical task in ensuring the security and resilience of computer systems and networks. Traditional techniques for malware assessment often rely on signature-based methods, which struggle to keep up with the constantly evolving landscape of malware variations. Recently, the application of advanced deep learning methods has shown promising results in automating the malware classification process. This study presents an innovative strategy for classifying malware images using the Vision Transformer (ViT) architecture. The ViT model is adapted to the domain of malware analysis by representing malware images as input tokens. A comprehensive dataset of 14,226 malware samples from 26 families was used to evaluate the effectiveness of this approach. A comparative analysis was performed between the performance of the ViT-based classifier, traditional machine learning approaches and other deep learning architectures. Our experimental results demonstrate the potential of ViT in handling malware images, achieving a classification accuracy of 98.80%. The presented approach establishes a strong foundation for further research in utilizing cutting-edge deep learning architectures for enhanced malware analysis and detection techniques.

Keywords: Malware detection, Vision Transformer, Deep learning, Network Security.

1. Introduction

The relentless advance of Internet technology has brought about a period of rapid expansion in the computer software sector. This has led to the development of a wide range of software applications that have become seamlessly integrated into the fabric of everyday life [1]. Nevertheless, this technological advancement has concurrently led to a concerning issue: the rampant proliferation of detrimental malware. This presents a substantial threat to the security of users' personal information, causing significant disruptions to computers, servers, and cloud infrastructures [2]. Malware is a type of software that threatens computer systems today and is designed to steal user data, exploit systems or for other malicious purposes [3]. This malware often runs without the user's permission or awareness and compromises personal security, privacy and the integrity of computer systems [4]. The most common types of malware today can be summarized as follows:

- ***Viruses:*** Malware that can infect other files and spread by making copies of themselves. They usually attach themselves to other files, infect them and then spread [5].
- ***Worms:*** Malicious software that self-replicates and spreads rapidly across computer networks to other systems. They do not infect files but spread by sending copies of themselves across the network to other devices [6].
- ***Trojans:*** Malicious software hidden in seemingly innocuous and useful programs that users tend to download. While installed by the user, its real purpose may be to covertly cause damage or steal information [7].
- ***Spyware:*** Software designed to secretly steal information by monitoring a user's computer activity. This information can often be sensitive, such as a user's online habits, personal or financial information [8].
- ***Adware:*** Malware designed to generate revenue by constantly displaying advertisements to the user. The ads appear without the user's consent and often have a negative impact on the user's experience [9].
- ***Ransomware:*** Malware that encrypts the user's files or system and demands a ransom to get them back [10].

Cite as: O. Katar and O. Yildirim, “Classification of malware images using fine-tunned ViT”. *Sakarya University Journal of Computer and Information Sciences*, vol. 7, no. 1, pp. 22-35, 2024. doi: 10.35377/saucis...1341082

Malware detection methods can be divided into two main categories: traditional static and dynamic approaches [11]. Static methods involve analyzing the structural characteristics of software to determine the presence of malware. In contrast, dynamic methods monitor the behavior of executing programs to identify potential malware instances [12]. These strategies offer distinct advantages and disadvantages in the quest for effective malware detection.

Dynamic detection is particularly accurate because it actively monitors the behavior of programs as they run, allowing it to quickly identify malicious software [13]. However, this approach is time-consuming because it requires continuous monitoring of running processes. This real-time analysis may not be conducive to the timely detection of emerging malware threats. In contrast, static detection can serve as a valuable complement to dynamic methods, helping to overcome their time-consuming aspect. By analyzing the structural attributes of software without executing it, static detection can quickly assess potential threats [14]. However, traditional static detection techniques rely on powerful antivirus engines and extensive virus databases [15]. This reliance on known signatures and patterns poses a significant challenge in detecting unseen or previously unknown malware, often resulting in the limited performance of traditional static approaches [16]. Efforts to strike a balance between static and dynamic detection techniques remain critical to improving the overall effectiveness of malware detection [17].

To overcome the limitations of traditional static detection approaches, researchers have looked at innovative ways to detect malware using visualization technology. These innovative techniques have shown promising performance in malware detection [18]. In many cases, malware variants are created through automation or reuse of critical function modules, resulting in a degree of similarity in their binary or assembly code [19]. Visualization technology is proving to be very useful in capturing these similarities and presenting them in a visual form. Interestingly, the challenges of malware detection are similar to those of image recognition, as both require the identification of variants or patterns within the original samples. By visually representing the structural characteristics and behavioral patterns of malware, visualization-based malware analysis reveals unique features that improve detection accuracy. By revealing hidden relationships and commonalities between malware variants, this approach promises to strengthen defenses against both known and previously unseen threats [20].

With the rapid advancement of artificial intelligence technology, researchers are increasingly using deep learning models to detect and classify malware. Yadav et al. [21] proposed a novel deep learning based two-stage framework for detecting and classifying DEX files images. The framework uses the EfficientNetB0 model to extract relevant features from malware images. These features are then processed through a stacking classifier, utilizing linear support vector machine (SVM) and random forest (RF) algorithms as base-level classifiers and logistic regression. The proposed method achieves impressive results, obtaining 100% accuracy in binary classification and 92.9% in 5-class classification. Khan et al. [22] proposed a malware detection framework called Deep Squeezed-Boosted and Ensemble Learning (DSBEL). The proposed DSBEL framework incorporates a novel Squeezed-Boosted Boundary-Region Split-Transform-Merge (SB-BR-STM) CNN that employs multi-path dilated convolutional, boundary and regional operations to capture global malicious patterns. The performance evaluation of the DSBEL framework and the SB-BR-STM CNN is performed on the IOT_Malware dataset, yielding results of 98.50% accuracy, 97.12% F-1 score, 95.97% recall and 98.42% precision. Xing et al. [23] proposed a state-of-the-art malware detection method. The method introduces a novel approach that involves representing malware as grey-scale images and incorporating an auto-encoder network for analysis. The viability of the grey-scale image representation is assessed by evaluating the reconstruction error of the auto-encoder. Furthermore, the dimensionality reduction capabilities of the auto-encoder are exploited to classify malware from benign software. Experimental evaluations conducted on an Android-side dataset demonstrate the effectiveness of the model, which achieves an impressive accuracy rate of 96% and a stable F1-score of around 96%. Asam et al. [24] introduced a novel CNN-based architecture called IoT malware detection architecture (iMDA) for effective detection. The iMDA architecture is designed with modularity and incorporates various feature learning schemes such as edge exploration and smoothing, multi-path dilated convolutional operations, and channel squeezing and boosting within the CNN framework. The performance evaluation of iMDA on a benchmark IoT dataset demonstrates its promising capabilities in malware detection, achieving 97.93% accuracy, 93.94% F-1 score, 98.64% precision, and 88.73% recall. Kumar and Janet [25] proposed deep transfer learning for malware image classification (DTMIC). By converting portable executable files (PEs) into grayscale images, DTMIC exploits the visual characteristics of similar malware families. The effectiveness and robustness of DTMIC are evaluated using MalImg and Microsoft BIG dataset. DTMIC achieves high detection accuracies of 98.92% for MalImg and 93.19% for Microsoft datasets, outperforming established CNN architectures.

Within the realm of malware detection and classification, CNN-based architectures, together with classical machine learning methods, have achieved significant success in image processing and are widely used in the literature. However, CNN-based classifiers often have limitations such as special architectural designs and input data with predetermined dimensions [26]. To overcome such problems, Vision Transformer (ViT) is a new approach that has recently attracted attention [27]. It is a transformer architecture based on this attention mechanism and designed for image classification problems. By treating the data as an irregular array of pixels, ViT provides a more flexible approach to better understand the relationships of objects in images and to detect important patterns [28]. ViT's particular scalability and ability to deal with large datasets and complex classification problems make it a suitable and promising candidate for malware image classification. Thanks to its ability to learn distant relationships between data, ViT can detect subtle differences between different types and subtypes of malware and achieve higher classification accuracy. In addition, ViT's attention mechanism prevents significant information loss in the feature extraction process, allowing for more comprehensive and detailed analysis.

This paper proposes a method for fine-tuning the default ViT architecture for automatic classification of malware images. The proposed model initially divides the image into patches and extracts features through the encoder network. Subsequently, these features are classified using an MLP (Multi-Layer Perceptron) head to determine the malware class. Moreover, to the best of the authors' knowledge, this is the first study on the classification of the MaleVis dataset utilizing the ViT model.

The main contributions of this study can be summarized as follows:

- The ViT model achieved high performance values using the MaleVis dataset, which contains 25 different types of malware.
- The ViT model demonstrated effective prediction on input images of different classes, eliminating the need for any feature engineering.
- A system that can classify malware images without depending on any resolution value.
- Leveraging the transfer learning method, the study attained higher performance values compared to similar research with fewer hardware requirements.

The remaining sections of this paper are structured as follows: Section 2 outlines the proposed method, explains the dataset used for model training, introduces the classifier model and describes the performance metrics. Section 3 explains the experimental setup and presents the results obtained, and Section 4 contains the discussion section of the study. The conclusions of the study are presented in Section 5.

2. Material and Methods

This paper proposes a ViT model for classifying malware images. Instead of taking the input images directly as input, the model processes them by dividing them into patches and vectorizing them. This approach allows the model to work faster and more efficiently by processing smaller parts of the input images by dividing them into patches. A schematic diagram of the proposed malware image classification method is given in Figure 1.

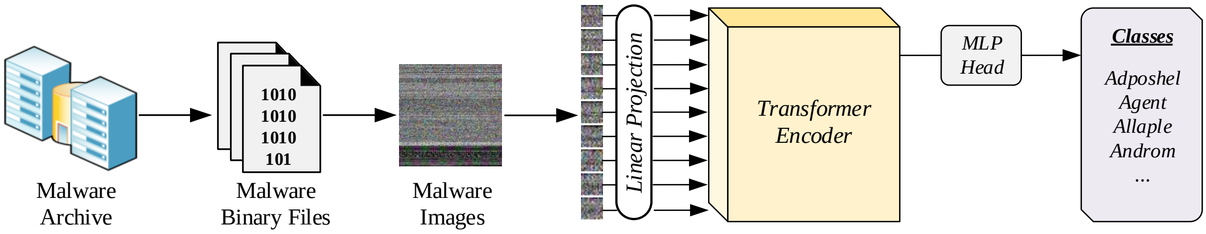


Figure 1 Schematic diagram of the proposed method.

The effectiveness of the deep learning models is highly dependent on the quality, diversity and size of the dataset. A well-curated and comprehensive dataset of malware images plays a crucial role in enabling the ViT model to learn meaningful representations and features that distinguish between different types of malware. By using a large and diverse dataset of malware images, the ViT model can effectively learn to extract relevant patterns and structures from the input images. The model's ability to divide the input images into smaller patches and vectorize them allows it to exploit the inherent spatial relationships in the dataset. This process facilitates the capture of fine-grained details and local features within each patch, enabling the model to make accurate and efficient classifications. Furthermore, the use of a diverse dataset can help improve the generalization capabilities of the ViT model. By exposing the model to a wide variety of malware samples with different characteristics, the model can better adapt to unseen and real-world scenarios. Consequently, this increases the overall robustness and reliability of the proposed malware image classification method.

2.1 Dataset

The public dataset used in this study is called Malware Evaluation with Vision (MaleVis) [29]. The MaleVis dataset consists of 25 different malware families, collected from samples that appeared between 2017 and 2018. These samples were created on PE files prepared by a cybersecurity company. These binary code files were converted into 224×224-pixel PNG images using the 'Bin2png' library. The samples that were randomly selected from the dataset are shown in Figure 2.

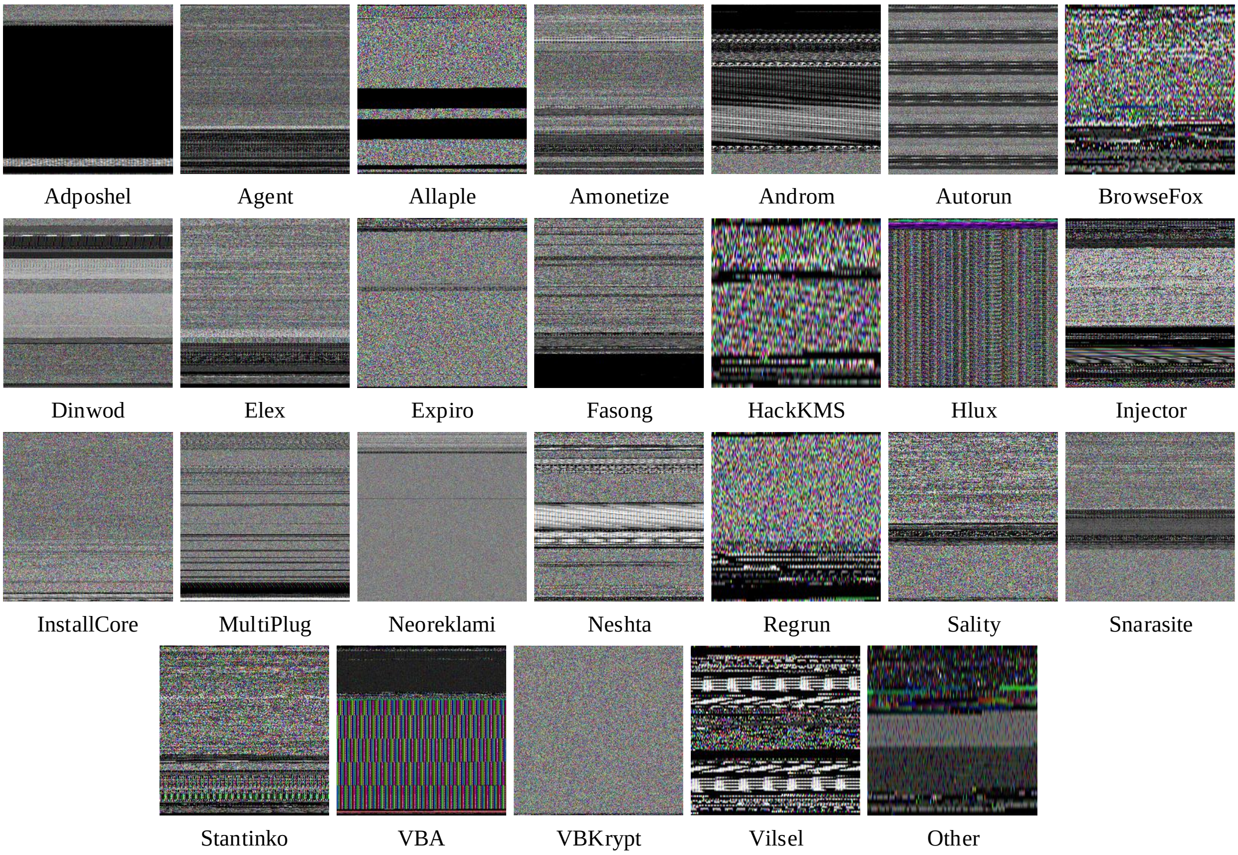


Figure 2 MaleVis Samples[29]

Table 1 Distribution of the dataset samples

Class Name	Category	Number of Samples in MaleVis	Number of Samples in Subsets (Train/Validation/Test)
Adposhel	Adware	494	396/49/49
Agent	Trojan	470	376/47/47
Allaple	Worm	478	382/48/48
Amonetize	Adware	497	397/50/50
Androm	Backdoor	500	400/50/50
Autorun	Worm	496	396/50/50
BrowseFox	Adware	493	395/49/49
Dinwod	Trojan	499	399/50/50
Elex	Trojan	500	400/50/50
Expiro	Virus	501	401/50/50
Fasong	Worm	500	400/50/50
HackKMS	Trojan	499	399/50/50
Hlux	Worm	500	400/50/50
Injector	Trojan	495	397/49/49
InstallCore	Adware	500	400/50/50
MultiPlug	Adware	499	399/50/50
Neoreklami	Adware	500	400/50/50
Neshta	Virus	497	397/50/50
Regrun	Trojan	485	389/48/48
Sality	Virus	499	399/50/50
Snarasite	Trojan	500	400/50/50
Stantinko	Backdoor	500	400/50/50
VBA	Virus	500	400/50/50
VBKrypt	Trojan	496	396/50/50
Vilsel	Trojan	496	396/50/50
Other	Legitimate	1832	1466/183/183

The MaleVis dataset, which consists of 26 classes, only possesses legitimate content in the form of the "Other" class, while the remaining classes, constitute malware. The dataset is divided into five categories of malware, including Adware, Trojan, Worm, Backdoor, and Virus. With a balanced distribution of approximately 500 samples per class, no data augmentation was deemed necessary. The dataset was divided into training, validation, and testing subsets, with 80% of the samples allocated for training, 10% for validation, and 10% for testing. The class-wise distribution of the dataset is presented in Table 1.

2.2 ViT Model

The profound impact of transformer networks on natural language processing tasks has been widely acknowledged. Building upon the success of the original transformer architecture, Dosovitskiy et al. [30] introduced the ViT model, specifically tailored for image processing tasks. The ViT model consists of self-attention blocks and MLP networks, equipped with linear projection and positional embedding mechanisms to handle input images effectively. The ViT architecture is based on the process of dividing the input image into fixed size, non-overlapping patches [31]. These patches are then flattened and a spatial embedding step is performed using linear projection. The purpose of spatial embedding is to preserve the spatial information of the original image within the flattened patches. The resulting vector is then fed into a stack of N transform encoder blocks. The architecture of these encoder blocks used for feature extraction in the ViT model is given in Figure 3.

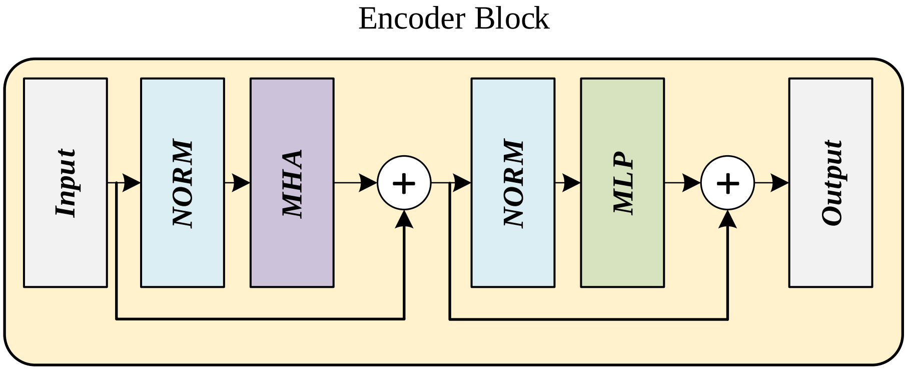


Figure 3 Architecture of the ViT encoder block

The basic components of a transformer encoder block include multi-head self-attention (MHA) and MLP layers. Each component is complemented by a normalization layer and a residual connection for improved training stability. Within MHA, self-attention is applied to each patch individually, producing three distinct vectors: query (Q), key (K) and value (V) [32]. To measure the importance or saliency of each embedded patch, a dot product operation is performed between the Q and K vectors, producing a score matrix. This matrix is then passed through the SoftMax activation function, which converts the scores into attention weights [33]. Finally, the output is obtained by element-wise multiplication of the attention weights and the V vector. This process produces the self-attention result as seen in Equation 1, where dk denotes the dimensionality of the key vector K.

$$Self\ Attention\ (Q, K, V) = SoftMax(\frac{QK^T}{\sqrt{d_k}}) \times V \tag{1}$$

The self-attention matrices are concatenated and then passed to a linear layer followed by a regression head. This application of self-attention allows the model to recognize relevant semantic features at different locations in the image, facilitating accurate classification. Within the transformer encoder, there can be MHA blocks, each contributing to the overall understanding of the image. Following the MHA layer, the transformer block contains an MLP. These MLP layers are equipped with a GeLU activation function [34]. The final output of the transformer block is calculated as shown in Equation 2. K.

$$Transformer\ Encoder_{out} = MLP(NORM(MHA_{out})) + MHA_{out} \tag{2}$$

2.3 Performance Metrics

Evaluating the performance of deep learning models is critical to understanding their effectiveness in solving real-world problems. Performance metrics play a vital role in quantifying the quality and reliability of these models, and some of the most important metrics include true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). These metrics can be summarized as follows.

- TP signifies the number of instances that are correctly classified as belonging to that specific class. It's the count of correctly identified positive samples for each class.
- TN signifies the number of instances that are correctly classified as not belonging to that specific class. It's the count of correctly identified negative samples for each class.
- FP signifies the number of instances that are incorrectly classified as belonging to that specific class when they actually don't belong to it. It's the count of incorrectly identified positive samples for each class.
- FN signifies the number of instances that are incorrectly classified as not belonging to that specific class when they actually belong to it. It's the count of incorrectly identified negative samples for each class.

In addition to the basic metrics of TP, TN, FP and FN, the evaluation of deep learning models includes the construction of a confusion matrix, which provides a comprehensive summary of the model's performance across all classes. The confusion matrix is a table that represents the predicted labels against the true labels, allowing a more detailed analysis of the model's behavior. The confusion matrix for the performance evaluation of a 26-class deep learning model is shown in Figure 4.

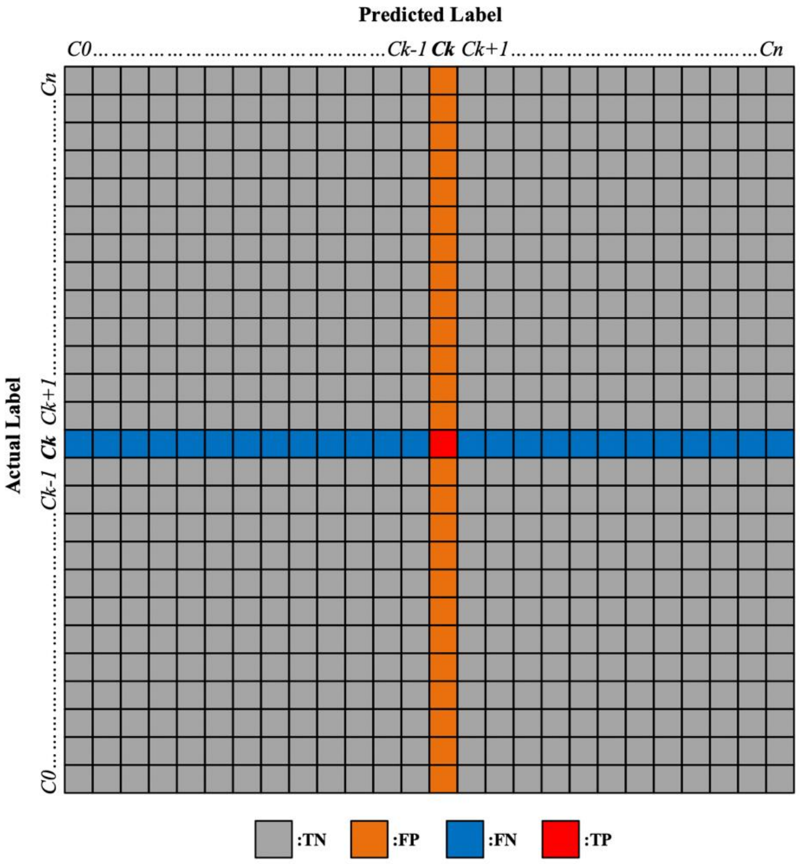

Predicted Label
C0……………………………………Ck-1 Ck Ck+1…………………………………Cn
Actual Label
C0……………………Ck-1 Ck Ck+1……………………Cn
:TN
:FP
:FN
:TP

Figure 4 A confusion matrix with 26 classes

The main performance measures and mathematical equations used to evaluate deep learning models can be summarized as follows:

1. ***Accuracy (Acc):*** This metric quantifies the overall correctness of the model's predictions. It is the ratio of correctly classified samples to the total number of samples. The Acc can be calculated as seen in Equation 3.

$$Acc = \frac{(TP + TN)}{(TP + FP + FN + TN)} \tag{3}$$

2. ***Precision (Pre):*** Premeasures the proportion of positively labelled samples that the model correctly identifies. It focuses on the ability of the model not to misclassify negative samples as positive, reflecting its ability to make accurate positive predictions. The Pre can be calculated as seen in Equation 4.

$$Pre = \frac{TP}{(TP + FP)} \tag{4}$$

3. ***Recall (Rec):*** Rec, also known as sensitivity, assesses the ability of the model to correctly identify true positive samples. It is particularly relevant when the aim is to minimize false negatives and avoid missing positives. The Rec can be calculated as seen in Equation 5.

$$Rec = \frac{TP}{(TP + FN)} \tag{5}$$

4. ***Specificity (Spe):*** This metric assesses the model's ability to correctly identify negative samples. It gauges the model's performance in avoiding false positives and accurately recognizing negative instances. The Spe can be calculated as seen in Equation 6.

$$Spe = \frac{TN}{(TN + FP)} \tag{6}$$

5. ***F-1 Score (F1):*** The F1 is the harmonic mean of Pre and Rec. It is particularly useful when there is an imbalance between positive and negative samples, as it balances the trade-off between precision and recall. The F1 can be calculated as seen in Equation 7.

$$F1\,Score = \frac{(2 \times Pre \times Rec)}{(Pre + Rec)} \tag{7}$$

3. Experiments

The experiments conducted to evaluate the performance of the ViT model are presented in this section. Additionally, the following sections present the analyses of the experimental results, along with the performance metrics.

3.1 Experimental Setups

In this study, we used the ViT-B/16 model, which has a resolution of 224×224 pixels and was pre-trained on the ImageNet [35] dataset. The ViT-B models include a hidden size of 768, an MLP size of 3072, and an overall parameter count of 86 million [30]. The input-output shapes and trainability of the layers of the model are summarized in Table 2.

Table 2 Details of the ViT model

Count	Layer	Input Shape	Output Shape	Trainable
×1	Patch Embedding	(1, 3, 224, 224)	(1, 196, 768)	True
×1	Dropout (pos_drop)	(1, 197, 768)	(1, 197, 768)	False
×1	Identity (patch_drop)	(1, 197, 768)	(1, 197, 768)	False
×1	Identity (norm_pre)	(1, 197, 768)	(1, 197, 768)	False
×12 (Encoder)	LayerNorm (norm1)	(1, 197, 768)	(1, 197, 768)	True
	Attention (attn)	(1, 197, 768)	(1, 197, 768)	True
	Identity (ls1)	(1, 197, 768)	(1, 197, 768)	False
	Identity (drop_path1)	(1, 197, 768)	(1, 197, 768)	False
	LayerNorm (norm2)	(1, 197, 768)	(1, 197, 768)	True
	Mlp (mlp)	(1, 197, 768)	(1, 197, 768)	True
	Identity (ls2)	(1, 197, 768)	(1, 197, 768)	False
	Identity (drop_path2)	(1, 197, 768)	(1, 197, 768)	False
×1	LayerNorm (norm)	(1, 197, 768)	(1, 197, 768)	True
×1	Identity (fc_norm)	(1, 768)	(1, 768)	False
×1	Dropout (head_drop)	(1, 768)	(1, 768)	False
×1	Linear (head)	(1, 768)	(1, 26)	True

The model was implemented using the timm library. During model training, 11,380 samples were randomly selected from the dataset, representing 80% of the total dataset samples. The remaining samples were divided equally to be used for the validation and testing phases. A visual representation of the proposed experimental setup framework is shown in Figure 5.

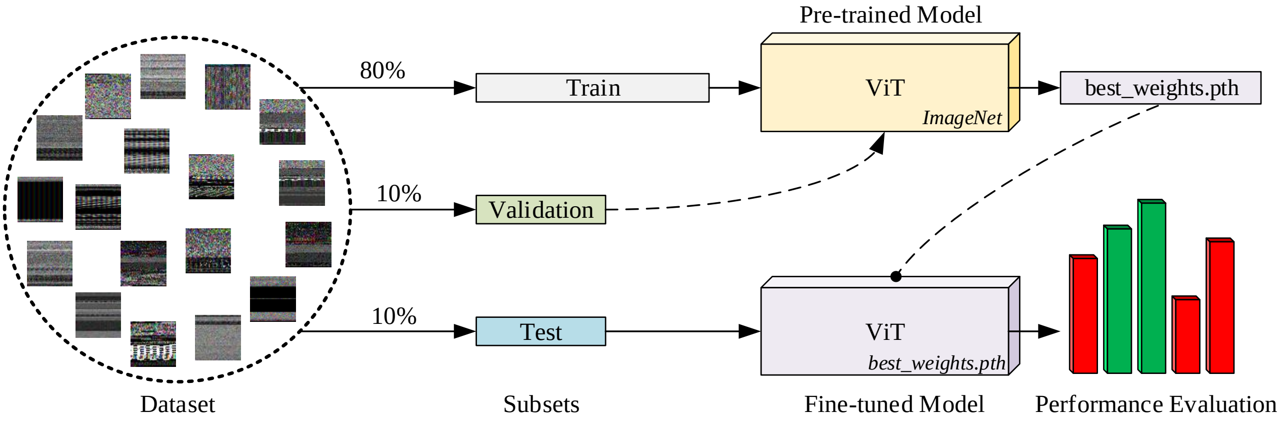


Figure 5 The proposed experimental setup framework

The training set was used to learn the parameters of the model, while the validation set was used to adjust the hyper-parameters and check for overfitting. The training and test samples were included in the training with a batch size of 32. The learning rate of the model was set to 0.00002 and the AdamW optimizer was used. The maximum number of epochs defined for the training and validation processes is 100. The CrossEntropyLoss function was used to calculate the loss value at each epoch. The software, hyperparameters and library versions used in the study are summarized in Table 3.

Table 3 Experimental environment and parameters

Name	Type	Version / Value
Python	Programming Language	3.10.11
Timm	Library	0.9.2
Torch	Library	2.0.1
Torchvision	Library	0.15.2
Transformers	Library	4.32.1
Batch Size	Hyperparameter	32
Learning Rate	Hyperparameter	0.00002
Max Epoch	Hyperparameter	100
Epsilon	Hyperparameter	0.000001

When training the ViT model with ImageNet weights, an early stopping function is defined to monitor the training phase. This function monitors the validation accuracy value and stops training if there is no improvement for twenty consecutive epochs. In this way, the weights of the epoch with the highest validation accuracy value were saved as 'best_weights.pth'. The best_weights.pth values were transferred to the ViT model, and the fine-tuned model was obtained. The fine-tuned ViT model was given test examples as input, which the model had never encountered before, and the performance of the model was determined by analyzing the predictions obtained.

3.2 Results

The ViT model was trained in the Google Colab environment using samples from the MaleVis dataset. By using optimized ImageNet weights as initial parameters, rather than random weights, the model achieved high accuracy rates in a relatively short time. The early stopping function set the validation accuracy value of 97.78% obtained in the 37th epoch as the stopping point, as there was no improvement in the subsequent twenty epochs. The weights obtained were saved for use during the test phase. The graphs showing the performance curves of the ViT model during the validation and training phases are shown in Figure 6.

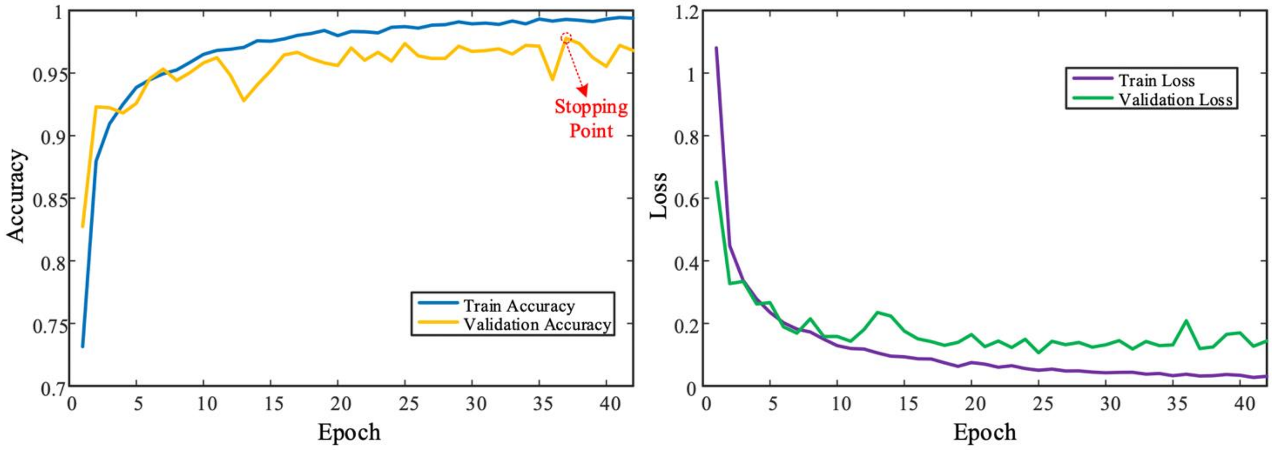
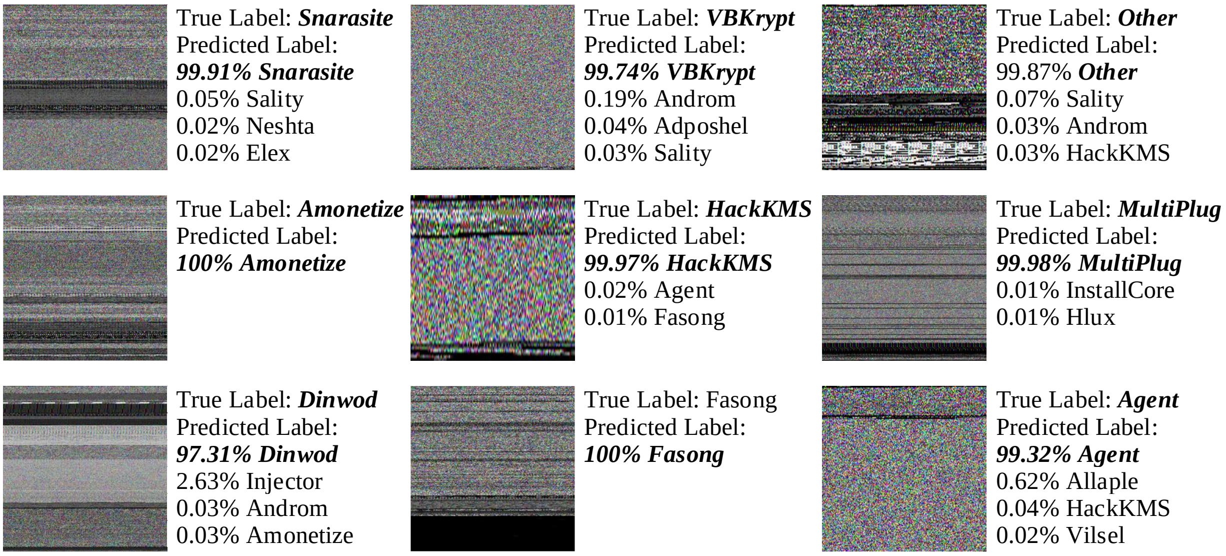


Figure 6 Graphs of the training and validation performance

Using the stopping point weights of the model, predictions were analyzed on randomly selected samples from the validation dataset. These predictions and their class-based probabilities are given in Figure 7.



True Label: ***Snarasite***
Predicted Label:
99.91% Snarasite
0.05% Sality
0.02% Neshta
0.02% Elex

True Label: ***VBKrypt***
Predicted Label:
99.74% VBKrypt
0.19% Androm
0.04% Adposhel
0.03% Sality

True Label: ***Other***
Predicted Label:
99.87% ***Other***
0.07% Sality
0.03% Androm
0.03% HackKMS

True Label: ***Amonetize***
Predicted Label:
100% Amonetize

True Label: ***HackKMS***
Predicted Label:
99.97% HackKMS
0.02% Agent
0.01% Fasong

True Label: ***MultiPlug***
Predicted Label:
99.98% MultiPlug
0.01% InstallCore
0.01% Hlux

True Label: ***Dinwod***
Predicted Label:
97.31% Dinwod
2.63% Injector
0.03% Androm
0.03% Amonetize

True Label: Fasong
Predicted Label:
100% Fasong

True Label: ***Agent***
Predicted Label:
99.32% Agent
0.62% Allaple
0.04% HackKMS
0.02% Vilsel

Figure 7 The validation predictions and their class-based probabilities

As the accuracy of the ViT model was at the desired level during training, the weights were transferred to the model. Test images were used as input to the model to verify the robustness of the model. The model achieved an accuracy rate of 98.80% with only 17 misclassifications on 1423 test images. The confusion matrix resulting from the model's predictions on the test images is shown in Figure 8. When analyzing the test results of the model, it can be seen that the predicted class for most of the misclassifications is 'Other'. The main reason for this is that the features of the 'Other' class, which contains more examples than other classes, are dominant.

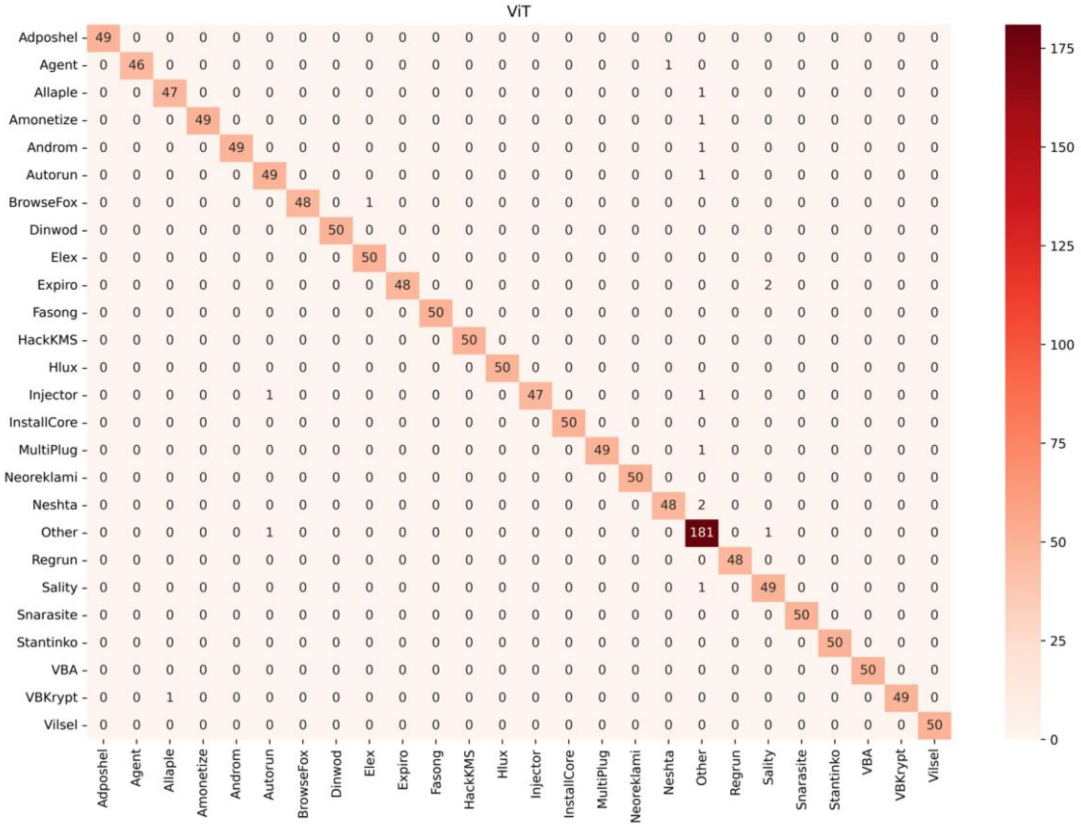


Figure 8 The confusion matrix for the test dataset

The block plots illustrating the class-based values of the performance metrics obtained during the test phase are presented in Figure 9.

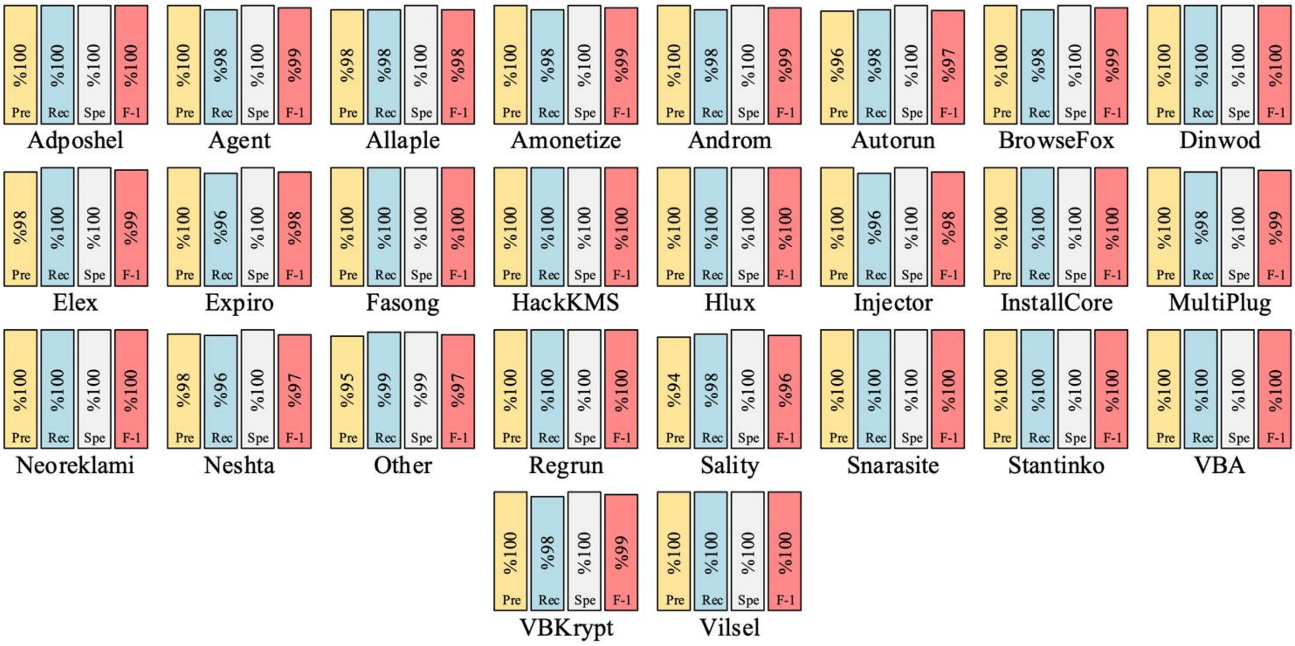


Figure 9 Class-based performance values obtained from the test samples

When analyzing the experimental results, it is observed that the Pre rate reaches 100% for twenty classes. Similarly, the remaining six classes are also characterized by high Pre values. The lowest Pre rate was observed in the Sality class with a value of 94.23%. Of the classes, only thirteen achieved a Rec rate of 100%, while the lowest recall value was obtained in the Injector class, with a rate of 95.91%. In terms of the Specificity metric, which boasts the highest average percentage, 25 classes reached a specificity of 100%. The Other class is the sole exception, exhibiting a specificity value that is distinctively lower than that of the other classes. Upon scrutinizing the class-based F-1 scores, it becomes evident that 25 classes exhibit scores surpassing 97%. However, the Sality class displays the lowest F-1 score at 96%. These statistical findings collectively demonstrate the model's proficient performance in the multiclass classification task, effectively identifying various classes.

4. Discussion

In the field of cybersecurity, the accurate classification of malware images is of paramount importance, serving as a central tool for identifying and mitigating a wide range of digital threats. While the classification of malware images has traditionally involved labor-intensive processes, the integration of machine learning methods to automate fundamental tasks has become essential in various domains, including information security. Leveraging advances in deep learning, a path pioneered by CNNs, the field continues to flourish with a number of novel architectures, each contributing to the evolving threat detection landscape. This study presents an implementation of ViTs, which have recently gained popularity, to classify malware with high accuracy. Table 4 lists similar research studies with the same dataset. Patil et al. [36] proposed a novel approach for malware image classification using machine learning, achieving accuracy rates of 93.00% for RF, 93.70% for EfficientNet-B0, and 92.00% for VGG-16 models. Ilyas and Mohammad [37] proposed a method for malware image classification. Their approach incorporated the employment of MobileNetV2, InceptionV3, ResNet50, and LittleVGG architectures. Notably, MobileNetV2 exhibited high performance compared to the other models, achieving an accuracy rate of 95.19%. Fathurrahman et al. [38] introduced a lightweight CNN model designed for malware image classification in IoT applications, particularly suitable for embedded systems. The proposed model achieved an average accuracy of 96.22%. Atitallah et al. [39] proposed a novel vision-based approach for classifying IoT malware images, utilizing deep transfer learning with ensembling strategies. The proposed approach, which fuses ResNet18, MobileNetV2, and DenseNet161 CNNs using an RF voting strategy, achieves exceptional performance with an accuracy of 98.68%.

Table 4 Comparison of our work with studies developed on the same dataset

Study	Year	Architecture	Model	Performance
Patil et al. [36]	2021	CNN	EfficientNet-B0	Acc = 93.70%
Iyas and Mohammad [37]	2021	CNN	MobileNetV2	Acc = 95.19%
Fathurrahman et al. [38]	2022	CNN	Custom CNN	Acc = 96.22%
Atitallah et al. [39]	2022	CNN	ResNet18+MobileNetV2+DenseNet161	Acc = 98.68%
The proposed study	2023	Transformer	ViT-B/16	Acc = 98.80%

In this study, the ViT model was used to classify malware images, achieving an impressive accuracy rate of 98.80% across 26 different classes. This performance surpasses the accuracy rates reported in other studies listed in Table 4. The ViT model's superiority can be attributed to its enhanced ability to comprehend pixel relationships, thanks to its attention mechanism, allowing it to effectively capture and represent crucial image features. However, in the context of malware classification, it's imperative to carefully consider the implications of both FP and FN classifications. FP can trigger unnecessary alarms or resource-intensive benign file investigations, while FN may pose significant security risks by allowing malicious files to evade detection. When examining the studies outlined in Table 4, it's evident that our research resulted in fewer FP and FN predictions. Additionally, the Sality class consistently displayed the lowest accuracy across all the studies in Table 4. By understanding the unique challenges associated with this class and exploring the potential reasons for its lower accuracy, can enhance the model's robustness and contribute to more reliable real-world malware detection systems. The ViT model's superior ability to grasp pixel relationships through its attention mechanism makes it especially advantageous for images with intricate structures, such as malware.

The advantages of our transformer-based model can be summarized as follows:

- The conceptual foundation of the proposed model is built upon ViTs, which are presently a prominent area of research. This is a pioneering work to investigate the performance of transformer-based image classifier models in the cybersecurity domain.
- Since the transfer learning method is used, the model achieves high performance values with low cost.
- The proposed model improves computational efficiency by vectorizing the input images with 16×16 patches, while minimizing important information lost in feature extraction.
- The proposed model can be fine-tuned and easily used to detect different types of malwares.

While our research demonstrates the encouraging possibilities of the ViT model in classifying malware images, it is important to acknowledge the inherent limitations and potential challenges associated with its application. Specifically, our ViT-based classifier requires a significant amount of labeled data for training, which may not be readily available in certain malware analysis scenarios, particularly for rare or emerging malware families. Furthermore, while the ViT model has impressive accuracy in classifying malware images, its real-time response time has not been evaluated. Another consideration is the computational requirements and scalability of the ViT model. Our study primarily focuses on its classification accuracy, but it's important to recognize that deploying the ViT model for real-time malware analysis on a large scale may demand substantial computational resources. Therefore, a thorough evaluation of the computational requirements and an assessment of the feasibility of deploying the ViT model for real-time malware analysis on a large scale are necessary. However, the black box nature of the proposed method may create difficulties in understanding how it makes classification decisions. Without the ability to gain insight into the features and attributes that the ViT model uses for classification, it may be difficult to gain meaningful information about the characteristics and behavior of different malware families. These limitations highlight the need for complementary methods or tools that can provide transparency and interpretability in the context of malware analysis, offering a more comprehensive and reliable approach to cybersecurity. In future work, we will extend our efforts by exploring malware detection using alternative state-of-the-art transformer-based architectures. Additionally, we intend to develop an explainable method that visualizes the specific pixel areas to which these models' pay attention in their predictions.

5. Conclusions

In this study, we proposed a ViT model designed for the automated classification of malware images. The proposed model is trained and validated on a public dataset with 26 different classes consisting of 14,226 samples. The ViT model with ImageNet weights is fine-tuned on malware images. As a result of the training phase using the early stopping function, the weights of the epoch with the highest validation accuracy value were recorded and implemented to the model. This model achieved 98.80% accuracy on test images it had never seen before. When analyzing the model's predictions on the test images, it can be seen that all performance metrics reach 100% for 12 classes. The performance values obtained for the other classes are also quite high. However, the model showed the lowest classification performance on the Sality class samples. The main limitation of the study is that it does not evaluate the performance of real-time applications. The proposed model can perform the classification of malware images automatically and can be effectively used by experts due to its high accuracy rates. Moreover, the proposed model can be easily fine-tuned for similar tasks to achieve high performance with low training costs.

References

[1] M. Wazid, A. K. Das, J. J. P. C. Rodrigues, S. Shetty, and Y. Park, "IoMT malware detection approaches: analysis and research challenges," *IEEE Access*, vol. 7, pp. 182459–182476, 2019.

[2] A. Chakraborty, A. Biswas, and A. K. Khan, "Artificial intelligence for cybersecurity: Threats, attacks and mitigation," *arXiv preprint arXiv:2209.13454*, 2022.

[3] 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," *Electronics (Basel)*, vol. 12, no. 6, p. 1333, 2023.

[4] C. S. Yadav *et al.*, "Malware analysis in iot & android systems with defensive mechanism," *Electronics (Basel)*, vol. 11, no. 15, p. 2354, 2022.

[5] M. Z. Hasan, M. Z. Hussain, and Z. Ullah, "Computer viruses, attacks, and security methods," *Lahore Garrison University Research Journal of Computer Science and Information Technology*, vol. 3, no. 3, pp. 20–25, 2019.

[6] B. S. Rawal, G. Manogaran, and A. Peter, "Malware," in *Cybersecurity and Identity Access Management*, Springer, 2022, pp. 103–116.

[7] P. M. Datta, "Cybersecurity threats: Malware in the code," in *Global Technology Management 4.0: Concepts and Cases for Managing in the 4th Industrial Revolution*, Springer, 2022, pp. 155–170.

[8] K. Geldenhuys, "Spyware: Spying on everything you do," *Servamus Community-based Safety and Security Magazine*, vol. 114, no. 10, pp. 15–17, 2021.

[9] M. Agrawal, K. D. S. Mann, R. Johari, and D. P. Vidyarthi, "Cyber Risks and Security—A Case Study on Analysis of Malware," in *International Conference on Innovative Computing and Communications: Proceedings of ICICC 2022, Volume 3*, Springer, 2022, pp. 339–349.

[10] S. Thakur, S. Chaudhari, and B. Joshi, "Ransomware: Threats, identification and prevention," *Cyber Security and Digital Forensics*, pp. 361–387, 2022.

[11] S. Li, Q. Zhou, R. Zhou, and Q. Lv, "Intelligent malware detection based on graph convolutional network," *J Supercomput*, vol. 78, no. 3, pp. 4182–4198, 2022.

[12] A. Razgallah, R. Khoury, S. Hallé, and K. Khanmohammadi, "A survey of malware detection in Android apps: Recommendations and perspectives for future research," *Comput Sci Rev*, vol. 39, p. 100358, 2021.

[13] N. Galloro, M. Polino, M. Carminati, A. Continella, and S. Zanero, "A Systematical and longitudinal study

of evasive behaviors in windows malware," *Comput Secur*, vol. 113, p. 102550, 2022.

[14] Q.-D. Ngo, H.-T. Nguyen, V.-H. Le, and D.-H. Nguyen, "A survey of IoT malware and detection methods based on static features," *ICT Express*, vol. 6, no. 4, pp. 280–286, 2020.

[15] Y. Yang *et al.*, "GooseBt: A programmable malware detection framework based on process, file, registry, and COM monitoring," *Comput Commun*, vol. 204, pp. 24–32, 2023.

[16] U. Zahoora, A. Khan, M. Rajarajan, S. H. Khan, M. Asam, and T. Jamal, "Ransomware detection using deep learning based unsupervised feature extraction and a cost sensitive Pareto Ensemble classifier," *Sci Rep*, vol. 12, no. 1, p. 15647, 2022.

[17] B. Y. Sathwara, "A hybrid approach based on boosting algorithm for effective android malware detection," *International Journal of Computing and Digital Systems*, vol. 13, no. 1, pp. 189–206, 2023.

[18] S. Venkatraman, M. Alazab, and R. Vinayakumar, "A hybrid deep learning image-based analysis for effective malware detection," *Journal of Information Security and Applications*, vol. 47, pp. 377–389, 2019.

[19] S. Alrabaee, M. Debbabi, and L. Wang, "A survey of binary code fingerprinting approaches: taxonomy, methodologies, and features," *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–41, 2022.

[20] H. Naeem *et al.*, "Malware detection in industrial internet of things based on hybrid image visualization and deep learning model," *Ad Hoc Networks*, vol. 105, p. 102154, 2020.

[21] P. Yadav, N. Menon, V. Ravi, S. Vishvanathan, and T. D. Pham, "A two-stage deep learning framework for image-based android malware detection and variant classification," *Comput Intell*, vol. 38, no. 5, pp. 1748–1771, 2022.

[22] S. H. Khan *et al.*, "A new deep boosted CNN and ensemble learning based IoT malware detection," *Comput Secur*, p. 103385, 2023.

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

[24] M. Asam *et al.*, "IoT malware detection architecture using a novel channel boosted and squeezed CNN," *Sci Rep*, vol. 12, no. 1, p. 15498, 2022.

[25] S. Kumar and B. Janet, "DTMIC: Deep transfer learning for malware image classification," *Journal of Information Security and Applications*, vol. 64, p. 103063, 2022.

[26] Z. Lu, S. Liang, Q. Yang, and B. Du, "Evolving block-based convolutional neural network for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–21, 2022.

[27] M. Chen *et al.*, "Searching the search space of vision transformer," *Adv Neural Inf Process Syst*, vol. 34, pp. 8714–8726, 2021.

[28] M. M. Naseer, K. Ranasinghe, S. H. Khan, M. Hayat, F. Shahbaz Khan, and M.-H. Yang, "Intriguing properties of vision transformers," *Adv Neural Inf Process Syst*, vol. 34, pp. 23296–23308, 2021.

[29] A. S. Bozkir, A. O. Cankaya, and M. Aydos, "Utilization and comparision of convolutional neural networks in malware recognition," in *2019 27th Signal Processing and Communications Applications Conference (SIU)*, IEEE, 2019, pp. 1–4.

[30] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[31] J. Wu, R. Hu, Z. Xiao, J. Chen, and J. Liu, "Vision Transformer-based recognition of diabetic retinopathy grade," *Med Phys*, vol. 48, no. 12, pp. 7850–7863, 2021.

[32] P. S. Thakur, P. Khanna, T. Sheorey, and A. Ojha, "Explainable vision transformer enabled convolutional neural network for plant disease identification: PlantXViT," *arXiv preprint arXiv:2207.07919*, 2022.

[33] Y. Wu, S. Qi, Y. Sun, S. Xia, Y. Yao, and W. Qian, "A vision transformer for emphysema classification using CT images," *Phys Med Biol*, vol. 66, no. 24, p. 245016, 2021.

[34] S. Illium, R. Müller, A. Sedlmeier, and C.-L. Popien, "Visual transformers for primates classification and covid detection," *arXiv preprint arXiv:2212.10093*, 2022.

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[36] S. Patil *et al.*, "Improving the robustness of ai-based malware detection using adversarial machine learning," *Algorithms*, vol. 14, no. 10, p. 297, 2021.

[37] I. Alodat and M. Alodat, "Detection of Image Malware Steganography Using Deep Transfer Learning Model," in *Proceedings of International Conference on Data Science and Applications: ICDSA 2021, Volume 2*, Springer, 2021, pp. 323–333.

[38] A. Fathurrahman, A. Bejo, and I. Ardiyanto, “Lightweight convolution neural network for image-based malware classification on embedded systems,” in *2021 International Seminar on Machine Learning, Optimization, and Data Science (ISMODE)*, IEEE, 2022, pp. 12–16.

[39] S. Ben Atitallah, M. Driss, and I. Almomani, “A novel detection and multi-classification approach for IoT-malware using random forest voting of fine-tuning convolutional neural networks,” *Sensors*, vol. 22, no. 11, p. 4302, 2022.

Conflict of Interest Notice

The authors declare that there is no conflict of interest regarding the publication of this paper.

Ethical Approval and Informed Consent

It is declared that during the preparation process of this study, scientific and ethical principles were followed, and all the studies benefited from are stated in the bibliography.

Availability of data and material

Not applicable.

Plagiarism Statement

This article has been scanned by iThenticate ™.