Journal of Computers Vol. 34 No. 1, February 2023, pp. 87-99
doi: 10.53106/199115992023023401007

Malware Family Classification Based on Vision Transformer

Jing Li[1], Xueping Luo[2*]

[1] Department of Computer Science and Engineering, Southwest Minzu University, Chengdu, China
li.wade707@outlook.com
[2] Department of Mathematics, Southwest University for Nationalities, Southwest Minzu University, Chengdu, China
Xueping luo2014@outlook.com

Received 10 May 2022; Revised 28 August 2022; Accepted 19 September 2022

Abstract. Cybersecurity worries intensify as Big Data, the Internet of Things, and 5G technologies develop. Based on code reuse technologies, malware creators are producing new malware quickly, and new malware is continually endangering the effectiveness of existing detection methods. We propose a vision transformer-based approach for malware picture identification because, in contrast to CNN, Transformer's self-attentive process is not constrained by local interactions and can simultaneously compute long-range mine relationships. We use ViT-B/16 weights pre-trained on the ImageNet21k dataset to improve model generalization capability and fine-tune them for the malware image classification task. This work demonstrates that (i) a pure attention mechanism applies to malware recognition, and (ii) the Transformer can be used instead of traditional CNN for malware image recognition. We train and assess our models using the MalImg dataset and the BIG2015 dataset in this paper. Our experimental evaluation found that the recognition accuracy of transfer learning-based ViT for MalImg samples and BIG2015 samples is 99.14% and 98.22%, respectively. This study shows that training ViT models using transfer learning can perform better than CNN in malware family classification.

Keywords: cybersecurity, deep learning, vision transformer, malware classification, grayscale images

1 Introduction

The growth of the Internet sector in recent years has benefited customers and significantly accelerated societal and national development, but it has also given rise to several network security concerns. According to a recent Accenture analysis, malware assaults have devastating economic repercussions; it is estimated that each attack [1] results in an average loss of $2.6 million. In response, a popular area of research has been how to successfully detect malware.

Malware detection techniques are classified into two types: static [2] and dynamic [3]. The extraction of static features does not require running the program to be tested. The static method does not need to run executable file samples and directly analyzes the bytecode sequences, opcode sequences, DLLs, APIs, and PE format structures of the samples. It is only necessary to determine the location of the corresponding data and the extraction method during the extraction process. The fastest way to ascertain a program's semantics is by static code analysis, however the most significant barrier to static inspection techniques is software encryption, shelling, and obfuscation. Real-time program execution in a sandbox, dynamic call tracking, and traffic analysis are all possible with dynamic analysis. Dynamic detection techniques are not limited by shelling and obfuscation and enable a more direct analysis of PE file behavior characteristics, but they are also not without drawbacks. Dynamic detection is less efficient and challenging to deploy, and some malware can detect the operating conditions or computing environment in which it is running and thus escape detection [4].

Most malware samples have structural differences but share some standard functionality, "Family" refers to this collection of malware [5]. To save costs associated with domain knowledge and feature extraction, scientists have developed a method of visualizing malware as part of their malware detection research, the reason for this is the visual similarity between variants within the same family. Unlike other traditional malware analysis techniques, the visual-based analysis does not require statically rich semantics or dynamic execution of binaries. Texture features can be extracted from malware images and used to train classifiers.

Inspired by the recent success of Transformer on numerous computer vision tasks, this paper presents a new scheme for malware detection based on vision converters, aiming to replace traditional CNN classification models. Our solution balances the malware dataset under study while utilizing transfer learning techniques, thus it

* Corresponding Author

does not necessitate a high level of cybersecurity experience. This strategy incorporates concepts and methods from malware image visualization methods. The malware samples are first represented as byte graph graphics, where each byte is equivalent to a pixel in a grayscale image. The created grayscale image is then sent into a vision converter so that useful characteristics can be extracted and the detection model can be trained. The malware is then categorized and examined using the trained model. Experiments are conducted on the BIG2015 as well as MalImg dataset to evaluate the proposed method. In summary, the main contributions of this work:

1. This study is the first malware image classification method using a vision transformer that detects malware samples without feature engineering, such as binary disassembly or reverse engineering.

2. We use transfer learning methods to efficiently and correctly classify unbalanced datasets. We also conducted extensive experimental tests using precision, recall, f1 score, confusion matrix, and other metrics to achieve more than 98% classification accuracy on the BIG2015 as well as MalImg datasets.

3. Our study compares the ViT model with other CNN models and shows that the ViT model offers superior accuracy for classification.

The remainder of the essay is structured as follows: The study relating to the visual classification of malware is presented in Section 2. The strategy suggested in this study as well as the dataset are both covered in detail in Section 3. The analysis of our experimental findings is presented in Section 4. In Section 5, we summarize our findings and discuss our next steps.

2 Related Work

Static analysis of malware has yielded many results in recent years, and researchers have discovered many kinds of static features through data mining techniques. Researchers have discovered many kinds of static features through data mining techniques, which provide a rich theoretical basis for current research. Among them, malware Since first presented in 2011, the malware's static image texture feature has been widely adopted. Since its debut in 2011, this functionality has been extensively used in numerous malware studies and has undergone continual improvement.

H. Naeem et al. [6] transformed the field information of malware into grayscale images and then extracted D-SIFT local features and GIST global features of grayscale images. They achieved 97.4% accuracy for 25 malware families using a machine learning SVM model. This approach innovates the traditional grayscale image by combining global and local features to achieve better model robustness and classification accuracy. H. Yakura et al [7] proposed to incorporate the attention mechanism into CNN models so as to extract the attention mechanism feature map of each class of malware family, through which the attention mechanism can filter the grayscale images with The attention mechanism can filter the highest value pixel regions in grayscale images and reduce the time of manual analysis.

X. Huang et al [8] proposed a combination of dynamic and static malware classification method. In order to fill the blue channel and red channel of the image, you must first extract the malware's byte frequency and byte stream into pixel points. Install the Windows operating system and Cuckoo Sandbox software in the virtual machine after adjusting the image's blue, green, and red channels. Using Cuc Cuckoo, one may gather reports from dynamic malware analysis, extract the API call sequences from the reports, and highlight the crucial APIs with more conspicuous colors. In the aforementioned red and blue channel map, the more significant colors are used to indicate the key APIs, while the cooler colors are used to indicate the less significant APIs. Using the VGG16 depth model for classification, the results show that the combination of static and dynamic classification results is 10% more accurate than using only static analysis.

The malware classification method proposed by Rezende et al [9] first converts malware to grayscale images, extracts the bottleneck features of malware (the feature map activated in the last layer before the fully connected layer) using a VGG16 network pre-trained on ImageNet through transfer learning, and then uses the features to train an SVM classifier. This method is able to reduce the training time and achieve 92.97% correct rate in the task of classifying 20 malware families.

Cui et al. [10] combined the bat algorithm with CNN architecture to deal with the multiclass imbalance problem while using data enhancement methods to increase the number of samples. The location structure of the malware itself is altered when image enhancement methods are applied to the malicious domain. This technique may lessen the importance of structural features that are crucial for malware classification because the final fully connected layer of the neural network used in their study is sensitive to positional features. According to the experimental findings, the validation set accuracy for the results of image enhancement alone is also the lowest, even lower than in the case of no processing at all.

3 Models and Methods for Malware Classification

3.1 Vision Transformer

Transformer was proposed by Google in 2017 in the literature [11] which brought great shocks to the field of natural language processing and was a landmark model. With the advancement of research, some recent papers [12-14] innovatively introduced Transformer technology across domains to computer vision tasks. The Image Transformer [12], released in 2018, first migrated the Transformer architecture to the computer vision domain. From 2019 to the present, Transformer-based vision models have evolved rapidly, with many noteworthy new results. For instance, Carion et al. [13] developed a novel object detection framework called DETR (detection Transformer) in May 2020, which was the first time Transformer was applied to the field of target detection. In order to investigate the GPT-2 [15] algorithm on images and the performance of unsupervised accuracy, Chen et al. [14] suggested the iGPT model in July 2020. The ViT (vision Transformer) model, a self-attentive mechanism-based picture categorization technique, was proposed by Dosovitskiy et al. in October 2020 [16]. It is the first effort to use Transformer instead of conventional convolution.

Fig. 1 depicts the model's whole end-to-end architecture. It is made up of three layers: an embedding layer, an encoder, and a final head classifier. The first step is to partition the image X into 9 patchs, respectively, $x_1, x_2, \ldots, x_9$, after that, flatten this 9 patchs. By linearly transforming the flattened vector x_i with the matrix W, the image encoding vector z_i is obtained. The calculation formula is shown in Equation 1.

$$z_i = W.x_i, i \in \{1, \ldots, 9\} . \quad \textbf{(1)}$$

The input coding vector is obtained by summing the image coding vector z_i and the class coding vector z_0 with the corresponding position coding, input the input encoding vector into Vision Transformer Encoder to get the corresponding output $o_i, i \in \{0, \ldots, 9\}$. Finally, the class coding vector o_0 is input to the fully connected neural network MLP to obtain the class prediction vector $\hat{y}$, and the cross entropy loss is calculated with the actual class vector y to obtain loss values and update the weight parameters of the model using the optimization algorithm.

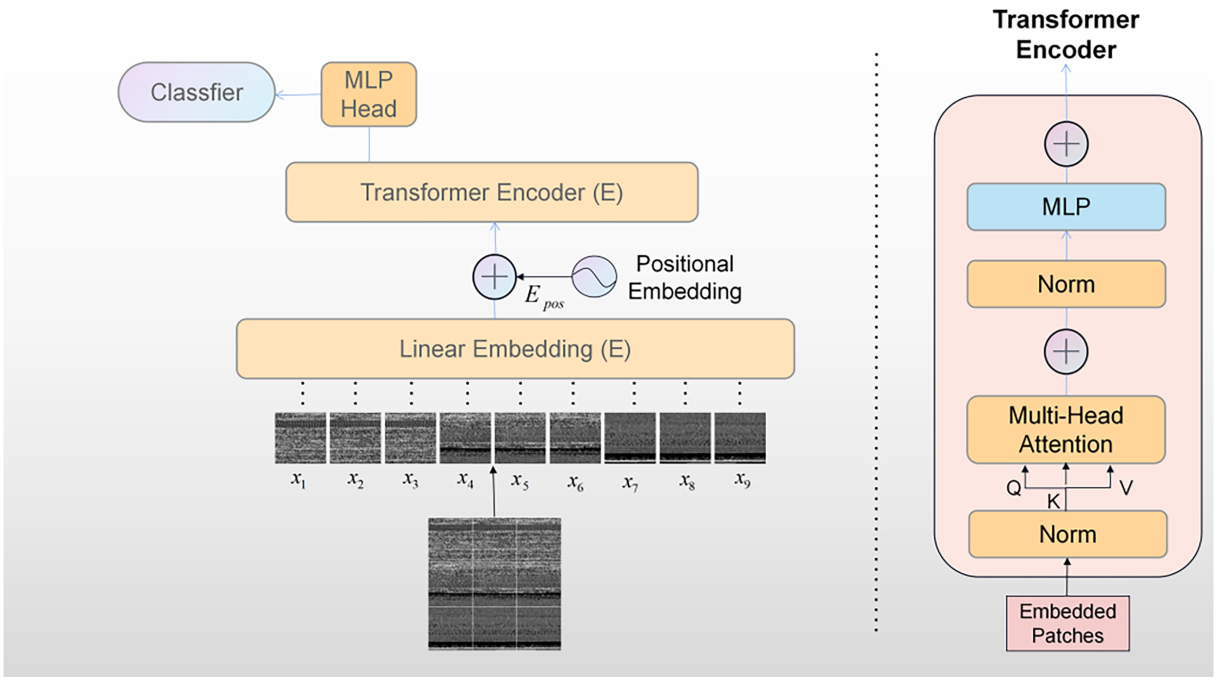


Fig. 1. The structure of vision Transformer

Positional Encoding. The order information of the words is naturally kept during processing because the typical RNN model handles the words in a phrase one at a time in a sequential pattern, therefore additional processing is not necessary. Transformer, on the other hand, loses the sequential information because each word in the phrase is entered into the network at the same time for processing. Transformer needs to be processed further in order to determine the relative position of each word. Positional encoding is one of the solutions, which includes adding an encoding that symbolizes the position information to the input so that the network is aware of the location and order of each word. In the literature [11], position encoding is introduced by the sine/cosine function as shown in Equation (2-3).

$$PE_{(pos,2i)} = \sin(pos / 10000^{2i/d}) . \tag{2}$$

$$PE_{(pos,2i+1)} = \cos(pos / 10000^{2i/d}) . \tag{3}$$

Where pos refers to the position of a feature at a moment in a sequence, d signifies the position encoding dimension, $2i$ denotes the even dimension, and $2i+1$ denotes the odd dimension (i.e., $2i \le d$, $2i+1 \le d$).

Self-Attention. To prevent the neural network from using recursion, the Transformer architecture adds a self-attentive mechanism. The mapping of the global interdependence between inputs and outputs is fully dependent on the self-attentive mechanism. In the literature, scaled dot-product attention is used [15]. Scaled dot-product attention is faster and more space-efficient than general attention since it uses dot products for similarity calculation. The input is linearly transformed to obtain the matrices Q (query), K (key value), and V (value), and the calculation formula is shown in Equation 4.

$$Attention(Q,K,V) = soft\max\left(\frac{QK^T}{\sqrt{d_k}}\right)V . \tag{4}$$

d_k denotes the number of columns of the matrices Q, K, commonly known as the vector dimension.

Multi-head Attention. The essence of the multi-headed attention mechanism is to split the query, key, and value multiple times while keeping the overall number of parameters constant. The query, key, and value parameters are split multiple times. Each set of split parameters is mapped to different subspaces of the high-dimensional space to calculate attention weights, thus focusing on different parts of the input. After multiple parallel calculations, the attention information in all subspaces is finally combined. The attention information in all subspaces is combined, and the formula is shown in Equation (4-5).

$$MultiHead\ (Q,K,V) = Concat\ (head_1, head_2, ..., head_h)W^o . \tag{5}$$

$$head_i = Attention\ (QW_i^Q, KW_i^K, VW_i^V) . \tag{6}$$

Where W^o, W_i^Q, W_i^K, and W_i^V are the parameter matrices in the linear transformation.

3.2 Dataset

MalImg dataset was developed by Vision Research Lab so that signal and image processing techniques could be assessed as malware detection and classification techniques. There are 9339 malware samples in the Vision Research Lab's collection from 25 families, they discuss how to convert malware binaries into images. This collection contains a variety of malware, including worms, backdoors, Trojans, and password stealers (PWS). Some of these instances are obfuscated, and the code has a polymorphic engine. The distribution of the number and names of each malware family in the MalImg dataset is shown in Fig. 2.

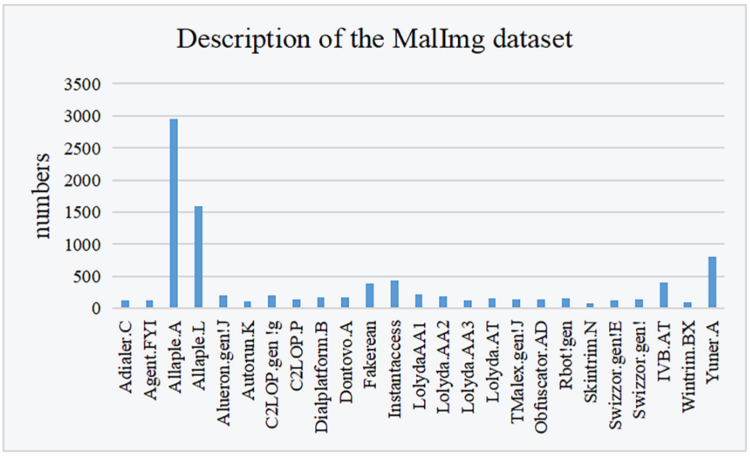


Fig. 2. Description of the MalImg dataset

Microsoft made the BIG2015 dataset available in 2015 as part of the kaggle challenge, which contains 21,741 samples from 9 malicious families, the training set with labels has 10,868 samples, whereas the test set without labels contains the remaining data. Two files are included in each malware sample: a hexadecimal representation of the binary file without the PE header and a metadata file created by the disassembly program IDA that contains the malware samples' machine code, assembly instructions, and other information. The distribution of the number and families of malware samples in the BIG2015 training set, which are utilized as the benchmark for model validation, is displayed in Fig. 3. This is because only the training set is labeled in BIG2015.

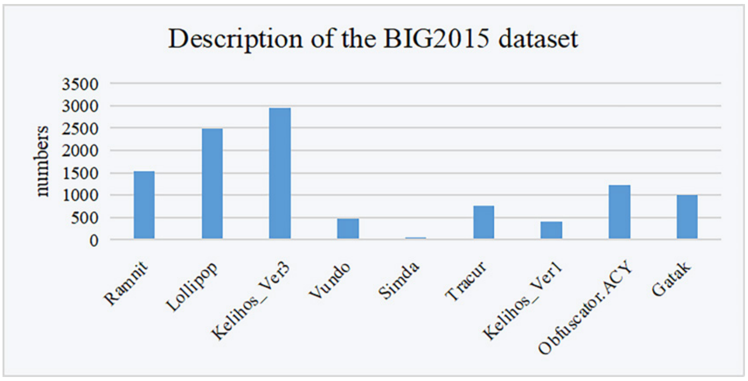


Fig. 3. Description of the BIG2015 dataset

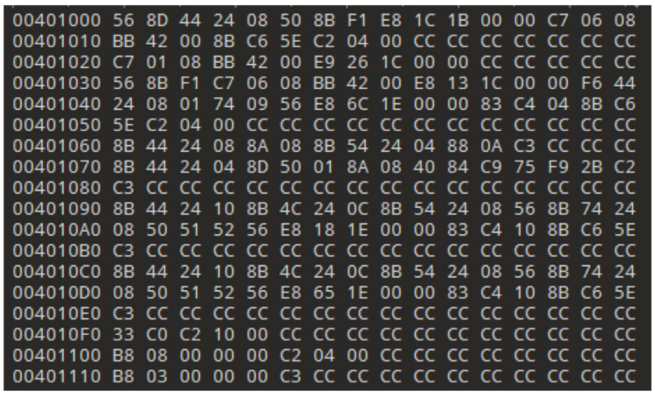
3.3 Pre-processing Dataset

Bin2Pixel algorithm is an algorithm proposed by Nataraj et al. [5] in 2011 to handle malware. It is currently the most widely used algorithm for malware visualization, which can convert malware into grayscale images of different sizes. The process is that the malware is opened in binary form with an 8-bit binary number as an unsigned integer as a pixel point, and the value of the pixel point ranges from 0 to 255, and the image height is adaptive according to the sample size to convert the sample into a grayscale image. The malware grayscale image width corresponds to Table 1.

Table 1. Image dimensions based on malware size

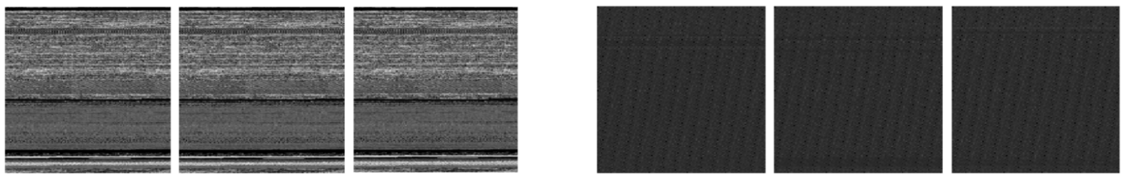
File size	Width (Pixels)
Less than 10kB	32
10kB-30kB	64
30kB-60kB	128
60kB-100kB	256
100kB-200kB	384
200kB-500kB	512
500kB-1000kB	768
More than 1000kB	1024

We use ".bytes" files and the Bin2Pixel algorithm to convert them into images. Fig. 4 depicts the binary format of the bytes file. Fig. 5 shows the similarity between different samples of the Adialer. C family and the Gatak family.



```
00401000 56 8D 44 24 08 50 8B F1 E8 1C 1B 00 00 C7 06 08
00401010 BB 42 00 8B C6 5E C2 04 00 CC CC CC CC CC CC CC
00401020 C7 01 08 BB 42 00 E9 26 1C 00 00 CC CC CC CC CC
00401030 56 8B F1 C7 06 08 BB 42 00 E8 13 1C 00 00 F6 44
00401040 24 08 01 74 09 56 E8 6C 1E 00 00 83 C4 04 8B C6
00401050 5E C2 04 00 CC CC CC CC CC CC CC CC CC CC CC CC
00401060 8B 44 24 08 8A 08 8B 54 24 04 88 0A C3 CC CC CC
00401070 8B 44 24 04 8D 50 01 8A 08 40 84 C9 75 F9 2B C2
00401080 C3 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC
00401090 8B 44 24 10 8B 4C 24 0C 8B 54 24 08 56 8B 74 24
004010A0 08 50 51 52 56 E8 18 1E 00 00 83 C4 10 8B C6 5E
004010B0 C3 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC
004010C0 8B 44 24 10 8B 4C 24 0C 8B 54 24 08 56 8B 74 24
004010D0 08 50 51 52 56 E8 65 1E 00 00 83 C4 10 8B C6 5E
004010E0 C3 CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC
004010F0 33 C0 C2 10 00 CC CC CC CC CC CC CC CC CC CC CC
00401100 B8 08 00 00 00 C2 04 00 CC CC CC CC CC CC CC CC
00401110 B8 03 00 00 00 C3 CC CC CC CC CC CC CC CC CC CC
```

Fig. 4. Bytes file fragment



(a) Smalpes of Adialer.C family (b) Smaples of Gatak family

Fig. 5. Samples of Adialer. C family and Gatak family

The code image texture resembles one another somewhat, as seen in the previous figure, because malware from the same family reuses the majority of the code, although the differences between dangerous code images of various families are very considerable. Therefore, based on this attribute, malware families that correlate to dangerous code grayscale images can be identified using image classification techniques.

3.4 Learning Style

Transfer Learning. Transfer learning is an approach for improving learning outcomes in new, related domains by applying previously learned knowledge from one domain to another that is connected to the first. Large datasets are frequently needed for traditional deep learning as training data, however gathering these datasets may be costly and time-consuming for small businesses and people. It's not always possible to get optimum training and testing results with a single and growing quantity of data. Models can frequently employ knowledge that has a proven track record of success through migration learning to achieve faster convergence, shorter training times, and improved test accuracy in new domains. Typically transfer learning is implemented in two ways, one is to directly freeze all layers of a trained model, making it untrainable, by which the output of a new sample is extracted as a feature description of the new sample. The other is based on a fine-tuning approach, which uses a new dataset to continue training the already trained model so that the model can be further adapted to the new data. The primary example diagram is shown in Fig. 6.

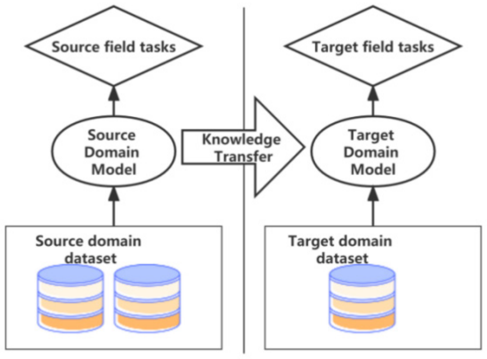


Fig. 6. Examples of transfer learning

Fine Tuning. Fine-tuning is a means to achieve migration learning [17], which is further training and tuning of the model based on an already trained model so that the model can output the desired results. It helps the model to adapt to new data in a shorter period [18] and has the advantage of being able to converge in a minimum number of epochs and being very friendly to small datasets than a pre-trained model. When pre-training is required to create a deep learning model The weights of the model that have been trained using a large amount of data can be used by fine-tuning when a large amount of data is not available. This significantly reduces the time needed for training and data collection, and the weights that are most frequently used are those based on ImageNet data training. The softmax layer of the model is transformed into the desired number of classifications, and then the weights from github are used. We then download the weights of the model without the fully connected layer from the weight repository of github.

It is clear post-analysis that the dataset used in this paper primarily categorizes malware images, in contrast to the ImageNet dataset. This paper does not freeze any layer of the model after loading the ImageNet weights so that the model can continue to be classified on the basis of the weights already loaded, despite the fact that the types of datasets in this paper are quite different from those of ImageNet and the classification tasks share a high degree of task similarity. Keep working out to achieve convergence. As a consequence, utilizing fine-tuning to apply model weights learnt in other domains to the dataset in this work should yield better results, and when the target dataset is significantly less than the source dataset, the fine-tuning helps to increase the model's generalization ability.

4 Experiment and Analysis

4.1 Experimental Environment

The studies were carried out on a Linux operating system using an Intel(R) Xeon(R) CPU E5-2670 v3 12-core CPU, 14GB RAM, a 128GB SSD, and an NVIDIA GeForce RTX 3060 with 12GB visual memory, and other experimental environments including Python3.8, Pytorch1.10, CUDA11.3. In addition, the model uses the matplotlib library to plot evaluation metrics for confusion matrices, accuracy scores, and classification reports.

4.2 Evaluation Metrics

In this paper, we classify the malware, and the most critical classification problem is the classification accuracy (Accuracy), Precision, Recall, and F1-Score are the most widely utilized measures.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad . \tag{6}$$

$$Precision = \frac{TP}{TP+FP} \quad . \tag{7}$$

$$Recall = \frac{TP}{TP + FN} \quad . \tag{8}$$

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \quad . \tag{9}$$

TP (True Positive) reflects the number of positive samples accurately predicted, FP (False Positive) represents the number of negative samples misclassified as positive samples, TN represents the number of negative samples correctly classified as negative samples, and FN (False Negative) represents the number of positive samples misclassified as negative samples.

4.3 Results and Analysis

Train and Test Results. ViT was built on the basis laid by Transformer, which was the first project to replace conventional convolution on huge datasets. It has made strides, but there are also unavoidable downsides. The Transformer model does not generalize the task successfully when there are inadequate data because it lacks inductive bias and does not have the translational invariance and local sensitivity of convolution. As a result, we selected the ViT-B/16 weight, which was initially trained on the ImageNet21K dataset before being adjusted.

In this paper, ViT-B/16 is modified by changing the output of its fully connected layer to 25 and 9, which represents the number of malware dataset family species in this paper, and the input picture resolution is set to $224 * 224 * 3$, which allows for a better understanding of the influence of different models on the same image categorization by adjusting. The fine-tuning technique in migration learning is also used in this model. In the model, the model is initially "knowledgeable". The starting LR is 0.01 and decreases with the number of repeats. Fig. 7 to Fig. 8 show the performance changes of the MalImg dataset and the BIG2015 dataset during the training of the model, the red line represents the training set, whereas the green line represents the test set. When Epoch is 10 or 12, we can observe that the model has converged on both datasets. Following training and testing, the accuracy of fine-tuning ViT-B/16 is 99.14 percent and 98 percent, respectively, while the loss is 0.6328 and 0.1044.

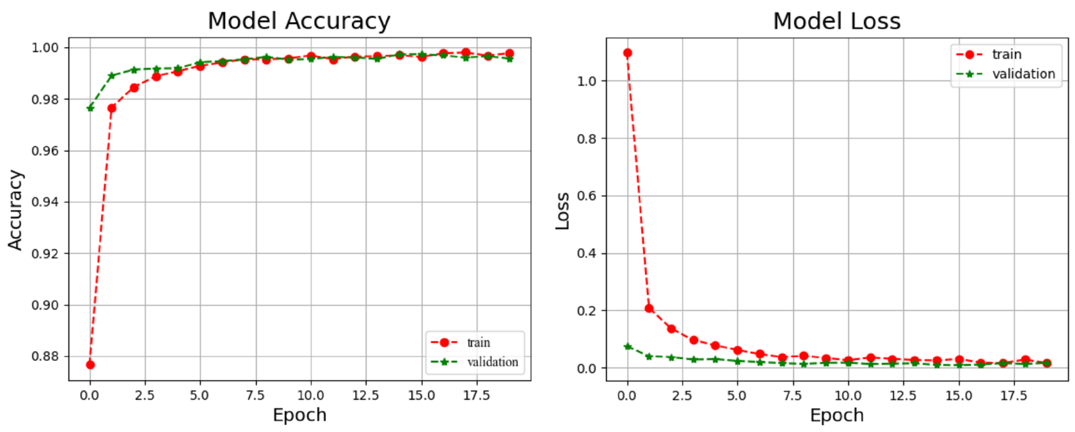


Fig. 7. Train and validation accuracy and loss for the MalImg dataset

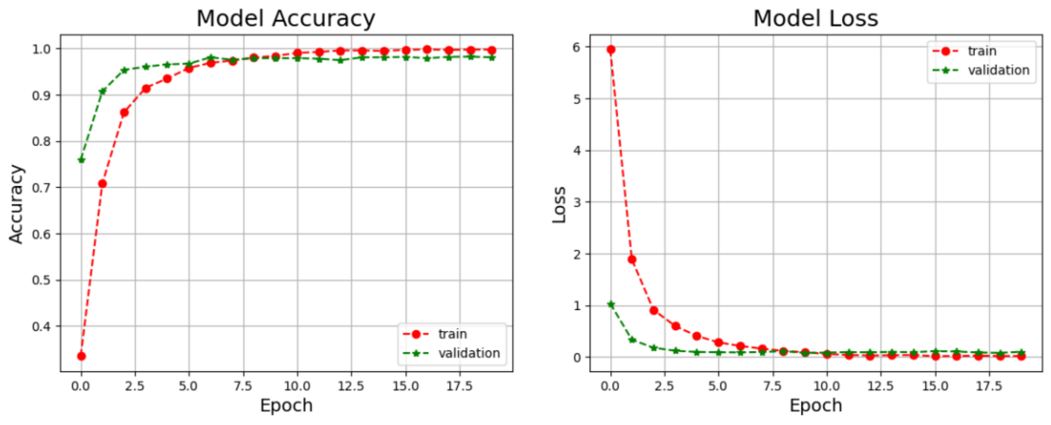


Fig. 8. Train and validation accuracy and loss for the BIG2015 dataset

To clearly see the model's classification details, Table 2 to Table 3 displays the classifier results for each malware family after applying the fine-tuned ViT-B/16 classifier, as well as the overall mean value for each evaluation measure. The results show that the proposed fine-tuned ViT-B/16 model achieves near-optimal significant values for all the evaluated metrics examined. Specifically, most of these features are correct, with an accuracy rate close to 98% or higher among the 34 malware families.

Table 2. Experiment results for MalImg dataset

Model	Precision	Recall	F1-score	Support
Adialer.C	1.0000	1.0000	1.0000	25
Agent.FYI	1.0000	1.0000	1.0000	25
Allaple.A	1.0000	1.0000	1.0000	125
Allaple.L	1.0000	1.0000	1.0000	100
Alueron.gen!J	1.0000	1.0000	1.0000	25
Autorun.K	1.0000	1.0000	1.0000	25
C2LOP.gen !g	0.9231	0.9600	0.9412	25
C2LOP.P	1.0000	0.9200	0.9583	25
Dialplatform.B	1.0000	1.0000	1.0000	25
Dontovo.A	1.0000	1.0000	1.0000	25
Fakerean	1.0000	1.0000	1.0000	75
Instantaccess	1.0000	1.0000	1.0000	75
LolydaAA1	1.0000	1.0000	1.0000	60
Lolyda.AA2	1.0000	1.0000	1.0000	25
sLolyda.AA3	1.0000	1.0000	1.0000	25
Lolyda.AT	1.0000	1.0000	1.0000	25
TMalex.gen!J	1.0000	1.0000	1.0000	25
Obfuscator.AD	1.0000	1.0000	1.0000	25
Rbot!gen	1.0000	1.0000	1.0000	25
Skintrim.N	1.0000	1.0000	1.0000	25
Swizzor.gen!E	1.0000	0.8000	0.8889	25
Swizzor.gen!I	0.8065	1.0000	0.8929	25
IVB.AT	1.0000	1.0000	1.0000	25
Wintrim.BX	1.0000	1.0000	1.0000	25
Yuner.A	1.0000	1.0000	1.0000	25

Table 3. Experiment results for BIG2015 dataset

Model	Precision	Recall	F1-score	Support
Ramnit	0.9461	0.9870	0.9661	231
Lollipop	0.9946	0.9892	0.9919	371
Kelihos_Ver3	1.0000	1.0000	1.0000	441
Vundo	0.9595	1.0000	0.9793	71
Simda	0.7143	0.8333	0.7692	6
Tracur	0.9817	0.9554	0.9683	112
Kelihos_Ver1	1.0000	1.0000	1.0000	59
Obfuscator.ACY	0.9773	0.9348	0.9556	184
Gatak	0.9800	0.9735	0.9767	151

However, As shown in the confusion matrix in Fig. 9 to Fig. 10, there were also some malware samples, including Swizzor.gen!E and Swizzor. gen!I, since both families have similar structures and patterns and are wrapped by the UPX wrapper, it is difficult for ViT to separate them. However, ViT still achieved more than 84% accuracy. As for the experiments on the BIG2015 dataset, most of the malware families can be accurately classified, except Simda, the identification accuracy can only reach 77% due to its tiny sample size.

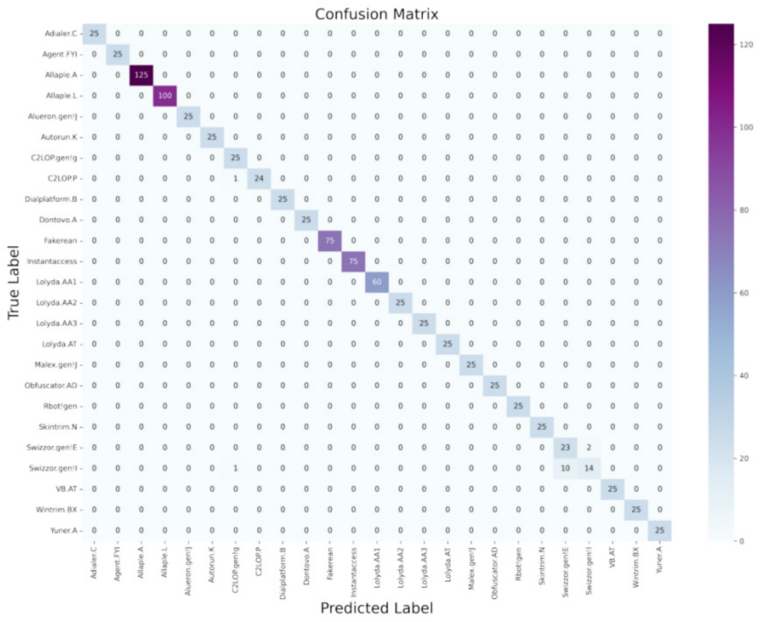


Fig. 9. Confusion matrix for the ViT-B/16 on the MalImg dataset

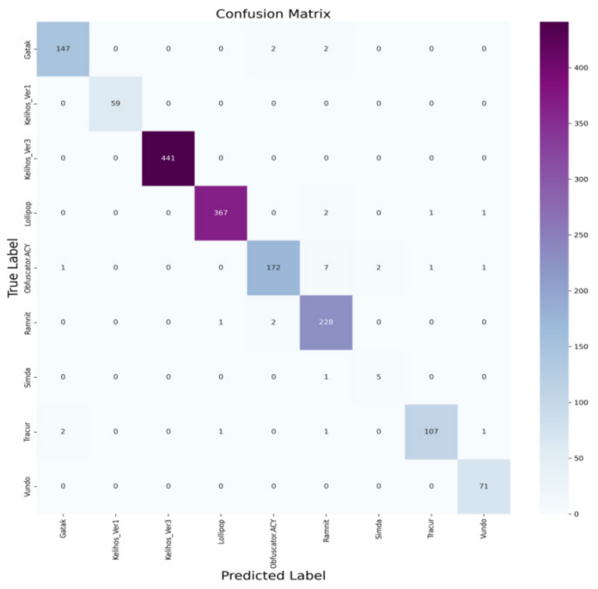


Fig. 10. Confusion matrix for the ViT-B/16 on the BIG2015 dataset

Comparison of Results for Different Architectures. We conducted many sets of comparison tests to illustrate the practicality of the suggested strategy in this research. To demonstrate the advantages of vision Transformer classification, fine-tuned CNN models are added for comparison on the above two datasets. In the realm of malware classification, ResNet152, BiT-L, EfficientNet-B7, and other models have been widely employed. These models are used to categorize the harmful dataset for comparison. We divide the dataset into a ratio of 8:2 for training as well as model evaluation. We used the adam optimizer and the softmax activation function for all deep learning architectures with a classification-cross-entropy loss function.

Table 4 displays the comparative trials of the four models using the ImageNet fine-tuning approach on the two datasets, and it can be seen that under the fine-tuning approach, when using pre-trained model weights on a large dataset like Imagenet21k, the ViT-B/16 model we used beats convolutional neural networks with the same migration learning in terms of malware identification performance. However, the classification performance on the BIG2015 dataset is weaker than that of the MalImg dataset for either model. Its accuracy is less than 99%, caused by its severely unbalanced dataset.

Compared with ViT-B/16, migration learning-based convolutional neural networks can also achieve excellent classification results, with the latest BiT-L model having an accuracy of even more than 98%. But ViT relies on an attention mechanism to capture global contextual information and can extract more powerful features. It is also pre-trained on large-scale datasets and can obtain better results than CNNs in image classification tasks.

Table 4. Experiment results for four different architectures

Model	MalImg dataset				BIG2015 dataset			
	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
ViT-B/16	0.9914	0.9928	0.9914	0.9915	0.9822	0.9826	0.9822	0.9822
BiT-L	0.9840	0.9844	0.9840	0.9834	0.9797	0.9804	0.9797	0.9798
EfficientNet-B7	0.9754	0.9782	0.9754	0.9738	0.9748	0.9763	0.9748	0.9751
ResNet152	0.9722	0.9789	0.9722	0.9679	0.9736	0.9742	0.9736	0.9736

Comparison of Results for Different Architectures. There are numerous additional deep learning architectures available for the categorization of malware. We also contrasted our outcomes with those from earlier studies, as shown in Table 5, to further assess the effectiveness of our proposed approach. These methods include traditional classification methods [22], deep learning methods [19-21, 23, 25] deep forest models [24], and our proposed method. The majority of models rely too heavily on feature generation, picture transformation, and data improvement, call for specialist algorithms, or involve some form of feature engineering approaches [19, 22, 24]. To deliberately pre-process data or generate or choose malware features calls for too much human involvement. These types of pipelines have become useless due to the exponential growth in the rate of newly generated malware. In the face of all this complexity, we propose a simple model that produces better results with almost no complexity. It takes advantage of visual converters that both mine long-range dependencies and compute in parallel, allowing it to outperform most other deep learning models for malware identification.

Table 5. Comparison with existing state-of-the-art methods on MalImg as well as on the BIG2015 dataset

Dataset	Methods	Accuracy (%)	Year & Reference
MalImg	GRU-SVM	84.92	(2019) [19]
	SPPNet	96.60	(2020) [20]
	LSTM	98.50	(2021) [21]
	Our method	99.14	
BIG2015	KNN-3	93.10	(2018) [22]
	CNN	97.50	(2019) [23]
	Deep Forest	97.20	(2020) [24]
	Alexnet + Resnet 50	96.50	(2021) [25]
	Our method	98.22	

Through comparison, it is easy to see that traditional machine learning algorithms have been difficult to achieve better results in the field of malicious image recognition, and the future research will focus more on the field of deep learning [20-21, 25]. Our method also has some drawbacks, ViT itself has the disadvantage of being computationally intensive and relying on large-scale datasets. Our dataset suffers from severe class imbalance, with some classes having less than 50 numbers and thus being divided into the wrong classes. In future research, we will explore the performance of the model in terms of efficiency, size, etc. In addition, we will use data augmentation as well as class balancing algorithms to improve the identification of malware.

5 Conclusion

The purpose of this work is to devise a technique for detecting malicious code that uses a vision Transformer. We chose two public datasets and the traditional Bin2Pixel algorithm is mainly used to directly transform malware into grayscale images, a process that does not require much domain expertise and is fast, accurate and at the same time resistant to obfuscation. The vision Transformer is used as a classification model for malware images. vision

Transformer model has achieved very good results in other image classification domains, but the model has not been well used in the malware classification domain. In this paper, we innovate the use of the model to classify images of malware. The model is trained using fine-tuning, it effectively utilizes the model's image categorization "knowledge" gained from other areas, and is capable of achieving high test accuracy on tiny dataset. We also compare other CNN models with fine-tuning. Our findings demonstrate that the vision Transformer model obtains greater accuracy than the CNN model by fine-tuning, further demonstrating that the ViT model can be well used in malware classification scenarios.

6 Acknowledgement

The authors express their acknowledgement for the anonymous review.

References

[1] S. Furnell, H. Heyburn, A. Whitehead, J.N. Shah, Understanding the full cost of cyber security breaches, Computer Fraud & Security 2020(12)(2020) 6-12.
[2] H. Kang, J. Jang, A. Mohaisen, H.-K. Kim, Detecting and classifying android malware using static analysis along with creator information, International Journal of Distributed Sensor Networks 11(6)(2015) 479174.
[3] W. Han, J. Xue, Y. Wang, L. Huang, Z. Kong, L. Mao, MalDAE: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics, Computers & Security 83(2019) 208-233.
[4] R.-J. Mangialardo, J.-C. Duarte, Integrating static and dynamic malware analysis using machine learning, IEEE Latin America Transactions 13(9)(2015) 3080-3087.
[5] L. Nataraj, S. Karthikeyan, G. Jacob, B.-S. Manjunath, Malware images: visualization and automatic classification, in: Proc. of the 8th International Symposium on Visualization for Cyber Security, 2011.
[6] H. Naeem, B. Guo, M.-R. Naeem, A light-weight malware static visual analysis for IoT infrastructure, in: Proc. 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD), 2018.
[7] H. Yakura, S. Shinozaki, R. Nishimura, Y. Oyama, J. Sakuma, Neural malware analysis with attention mechanism, Computers & Security 87(2019) 101592
[8] X. Huang, L. Ma, W. Yang, Y. Zhong, A Method for Windows Malware Detection Based on Deep Learning, Journal of Signal Processing Systems 93(2-3)(2021) 265-273.
[9] E. Rezende, G. Ruppert, T. Carvalho, A. Theophilo, F. Ramos, P. de Geus, Malicious software classification using VGG16 deep neural network's bottleneck features, in: Proc. 15th International Conference on Information Technology -New Generations, 2018.
[10] Z. Cui, F. Xue, X. Cai, Y. Cao, G. Wang, J. Chen, Detection of malicious code variants based on deep learning, IEEE Transactions on Industrial Informatics 14(7)(2018) 3187-3196.
[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.-N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Proc. of Advances in Neural Information Processing Systems 30 (NIPS 2017), 2017.
[12] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, S. Zagoruyko, End-to-end object detection with transformers, in: Proc. of European Conference on Computer Vision, 2020.
[13] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P.-H. Torr, L. Zhang, Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers, in: Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021.
[14] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, I. Sutskever, Generative Pretraining From Pixels, in: Proc. of the 37th International Conference on Machine Learning, PMLR, 2020.
[15] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, OpenAI Blog 1(8)(2019) 1-9.
[16] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale, in: Proc. of The International Conference on Learning Representations, ICLR 2021, 2021.
[17] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NaacL-HLT), 2019.
[18] M.M. Ghazi, B. Yanikoglu, E. Aptoula, Plant identification using deep neural networks via optimization of transfer learning parameters, Neurocomputing 235(2017) 228-235.
[19] H. Naeem, B. Guo, M.R. Naeem, D. Vasan, Visual Malware Classification Using Local and Global Malicious Pattern, Journal of Computers (Taiwan) 30(6)(2019) 73-83.
[20] R.-J. Maulana, G.-P. Kusuma, Malware Classification Based on System Call Sequences Using Deep Learning, JASTES Journal 5(4)(2020) 207-216.
[21] N. Marastoni, R. Giacobazzi, M.D. Preda, Data augmentation and transfer learning to classify malware images in a deep

learning context, Journal of Computer Virology and Hacking Techniques 17(4)(2021) 279-297.
[22] J. Fu, J. Xue, Y. Wang, Z. Liu, C. Shan, Malware visualization for fine-grained classification, IEEE Access 6(2018) 14510-14523.
[23] D. Gibert, C. Mateu, J. Planes, R. Vicens, Using convolutional neural networks for classification of malware represented as images, Journal of Computer Virology and Hacking Techniques 15(1)(2019) 15-28.
[24] S.-A. Roseline, S. Geetha, S. Kadry, Y. Nam, Intelligent vision-based malware detection and classification using deep random forest paradigm, IEEE Access 8(2020) 206303-206324.
[25] A. Darem, J. Abawajy, A. Makkar, A. Alhashmi, S. Alanazi, Visualization and deep-learning-based malware variant detection using OpCode-level features, Future Generation Computer Systems 125(2021) 314-323.