IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 June 2023, accepted 4 July 2023, date of publication 10 July 2023, date of current version 13 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3293530

RESEARCH ARTICLE

Global-Local Attention-Based Butterfly Vision Transformer for Visualization-Based Malware Classification

MOHAMAD MULHAM BELAL AND DIVYA MEENA SUNDARAM
School of Computer Science and Engineering, VIT-AP University, Amaravati 522237, India
Corresponding author: Divya Meena Sundaram (divyameena.s@vitap.ac.in)

ABSTRACT In recent studies, convolutional neural networks (CNNs) are mostly used as dynamic techniques for visualization-based malware classification and detection. Though vision transformer (ViT) proved its efficiency in image classification, a few of the earlier studies developed a ViT-based malware classifier. This paper proposes a butterfly construction-based vision transformer (B_ViT) model for visualization-based malware classification and detection. B_ViT has four phases: (1) image partitioning and patches embeddings; (2) local attention; (3) global attention; and (4) training and malware classification. B_ViT is an enhanced ViT architecture that supports the parallel processing of image patches and captures local and global spatial representations of malware images. B_ViT is a transfer learning-based model that uses a pre-trained ViT model on the ImageNet dataset to initialize the training parameters of transformers. Four B_ViT variants are experimented and evaluated on grayscale malware images collected from MalImg, Microsoft BIG datasets or converted from portable executable imports. The experiments show that B_ViT variants outperform the Input Enhanced vision transformer (IEViT) and ViT variants, achieving an accuracy equal to 99.49% and 99.99% for malware classification and detection respectively. The experiments also show that B_ViT is time effective for malware classification and detection where the average speed-up of B_ViT variants over IEViT and ViT variants are equal to 2.42 and 1.81 respectively. The analysis proves the efficiency of texture-based malware detection as well as the resilience of B_ViT to polymorphic obfuscation. Finally, the proposed B_ViT-based malware classifier outperforms the CNN-based malware classification methods in well.

INDEX TERMS Vision transformer, global attention, local attention, malware classification, visualization-based malware classifier, parallel processing.

I. INTRODUCTION

Malware is malicious software created with the intent of causing harm to a computer system, network, or cloud. This can include viruses, worms, trojan horses, spyware, ransomware, and more. Malware is usually spread through malicious emails, malicious websites, and malicious downloads [1]. For example, business email compromise (BEC) is an increasingly common form of cyber attack which targets businesses, organizations, and governments, according to Symantec's threat report (2020) [2]. LokiBot is a malicious piece of software that is designed to steal passwords, cryptocurrency wallets, and other sensitive data by placing malicious code into files that are hosted on cloud-based services. It was first discovered in 2019 and is believed to be related to other malicious programs like TrickBot and Azorult. LokiBot is also capable of evading detection by using encrypted communication and hiding its malicious code within legitimate applications [3]. Ryuk is a dangerous form of ransomware that can be used to extract large sums of money from victims. It typically targets large organizations, encrypting all of their data and demanding a ransom in exchange for its release. Ryuk is also known for its ability to spread quickly [4]. Malware detection techniques can

The associate editor coordinating the review of this manuscript and approving it for publication was Diana Gratiela Berbecaru.

be classified into two categories, each providing a distinct perspective. The first classification, known as static, dynamic, and hybrid, focuses on the analysis approach employed [5]. The second classification, signature-based and anomaly-based, focuses on the detection strategy utilized [6]. The analyzing techniques for malware detection are classified into 3 methods: static, dynamic, and hybrid. Static malware detection involves analyzing the code and metadata of malware such as size, developer information, number of downloads, and other relevant details to identify malicious behavior. It is a quick initial step that can detect suspicious code without execution. However, code obfuscation and reverse-engineering make it time-consuming and pose challenges in identifying threats [5]. In contrast, Dynamic malware detection uses controlled execution of malicious code to analyze its behavior and capabilities. It provides detailed insights into the code's actions and risks. Dynamic analysis is often used as a second step in malware detection. However, it can produce false positives, reducing detection accuracy. [5]. Hybrid malware detection combines both static and dynamic methods [5]. The most common approaches that are used to detect malware are signature-based methods [7]. Signature-based malware detectors rely on a database of known malware to identify and prevent malicious activities. They compare file characteristics to known signatures and take action if a match is found. It is an effective method to detect and prevent malware threats [8]. Signature-based malware detection has limitations. It cannot detect new or modified threats and can be computationally intensive due to processing large amounts of data [8]. Anomaly-based malware detectors identify malicious activities by detecting deviations from normal behaviour, aiming to detect unknown or zero-day malware that may evade traditional signature-based methods. These techniques employ machine learning algorithms [8]. Though machine learning-based malware detectors are effective and have good accuracy, the features should be manually and carefully collected by subject matter experts to make sure that the ML-based detectors would perform well [9]. Also, it may generate false positives or miss sophisticated attacks that mimic normal behaviour [7].

Recently, image visualization-based malware detection is becoming increasingly popular as it effectively detects malicious activities on a computer system. It is an effective way to detect sophisticated and complex malware that can bypass traditional methods of detection [5]. Instead of focusing on code analysis or behavioural patterns, this approach leverages the visual representation of malware samples to uncover potential malicious activities. The executable files or network traffic are converted into grey-scale or RGB images, and then these images are analyzed by the classification model to decide whether malware or not using visual features [10]. Image visualization-based malware detection has several advantages over traditional security techniques. First, it is more accurate in detecting malicious activities. Second, it is faster and easier to use than traditional methods. Lastly, it is more cost-effective than traditional methods, as it requires less human resources and time to set up and maintain [9].

Most of the recent studies focus on using convolutional neural networks (CNNs) as dynamic techniques for visualization-based malware detection [1], [5], [11], [12]. CNNs can analyze the visual representation of malware codes which are collected from executable files or network traffic, and identify malicious patterns based on the features that are extracted from the input image. However, CNNs are limited in their ability to identify subtle differences between malware variants [12]. As a result, they may not be able to detect small changes in malicious code that could affect the way the malware behaves [12].

The transformers-based approaches were only applied in natural language processing applications [13]. Some transformer-based approaches like Vision Transformer (ViT) [13] are currently applied in image classification tasks such as plant diseases classification [14], [15], chest X-ray image classification [16], etc. Though ViT outperforms the CNNs in terms of accuracy and image processing such that the spatial information of the image is retained in ViT [17], a few of the earlier studies developed a ViT-based malware classifier. However, some limitations should be tackled in ViT architecture to improve its performance in malware classification. ViT is able to process images at a much higher resolution than traditional CNNs, but processing very large images can still be computationally intensive. From the shallow layers of ViT, only the global representation of the input image is obtained but local representation should be also considered. it also requires large amounts of data in order to be trained accurately. Additionally, it can be computationally expensive, requiring powerful hardware and long training times [17].

This paper builds a B_ViT butterfly construction-based vision transformer model for visualization-based malware classification and detection. The main contributions of B_ViT are summarized as follows:

- B_ViT is an enhanced ViT architecture captures the local spatial representation and global spatial representation of malware images.
- All the variants of B_ViT such as BViT/B16, BViT/B32, BViT/L16, and BViT/L32 are experimented and evaluated on the MalImg [9], Microsoft BIG [18] datasets, and top-1000 PE imports. Then, they are compared with the performance of IEViT and ViT.

The following sections in this paper are organized as follows: Section II describes the related works and classifies the malware detection and classification approaches as per their functions. Section III discusses the grayscale malware image generation and the architecture of the proposed model B_ViT in detail. The materials and datasets are discussed in section IV. Section V analyzes the robustness of the proposed model to obfuscation. The performance analysis of B_ViT and the comparative analysis with the

other ViT-based architectures are presented in section VI. In addition, section VI shows a parallel analysis of B_ViT over IEViT, and ViT. A discussion and results interpretation are provided in section VII. Finally, the conclusion and limitations are provided in section VIII.

II. RELATED WORK

Static malware detection techniques are used to detect malicious software (malware) without actually executing it. This type of detection is based on analyzing the code statically or other static characteristics of the malware. Naik et al. [19] proposed a static malware detection method namely, fuzzy-import hashing. This method relies on two combined hashing techniques: fuzzy and import, each can be cooperative to improve the rate of malware detection. Liangboonprakong and Sornil [20] used N-grams sequential pattern features to build a malware classification approach. The N-grams are extracted by disassembling the binary executable into a hexadecimal string. The next step is sequential pattern extraction, which looks for frequently occurring sequences and uses them to build a feature vector to classify data. Narouei et al. [21] proposed a static malware detection technique to detect malware accurately and resist packing and injection of malware into legitimate software. Avdiienko et al. [22] mined the data flow for benign android apps, and used them to detect the android malicious apps. MUDFLOW tool is developed to mine and classify the dataflow in apps using FlowDroid. Static malware detection techniques are useful in terms of performance analysis due to identifying suspicious code without having to run it. However, these techniques can not resist code obfuscation or code packing besides, they require domain experts and reverse engineering [5].

Dynamic malware detection techniques are used to detect malicious code as it is running and can provide insights into the code's behavior within the operating system such as system calls and system resources. These behavioral features collected from the malware behavior can be used to implement machine learning or deep learning-based frameworks for malware detection [5]. Li et al. [23] proposed a DMalNet dynamic malware detection technique based on API semantic features and graph learning. The semantic features are extracted from API arguments and names by a hybrid encoder. The relationship between API calls is then converted into the graph's structural data using an API call graph that is derived from the API call sequence. Li et al. [24] implemented a malware detection framework using deep learning models that extract intrinsic features by capturing and combing significant features. API calls-based detection approaches have good accuracy in malware detection but their complexity is high and there is hardship to be used in a real-time environment that needs high capabilities. Recently, many state-of-arts suggested using hardware-based detection approaches that enhance the performance of model or framework [11]. Tian et al. [11] proposed MDCHD, a detection approach based on the control flow of target software at run-time that is collected from Processor Traces. Chen et al. [25] used the control flow traces of the target software to implement a malicious software deep learning-based detector. For minimal overhead execution tracing, Chen et al. [25] used processors with Intel Processor Trace enabled. There are some limitations in dynamic malware detection techniques such as selecting a proper environment to execute the software in it, the behavior of malware changing, or generating false sequences. Dynamic analysis-based methods are also non-scalable because they need a lot of resources and are environment-specific [5].

In recent studies, visualization-based malware detection is popular for complex and sophisticated malware detection that can bypass traditional malware detection techniques [5]. These methods are accurate, cost-effective, and require no feature extraction and fewer human resources [9]. Falana et al. [26] proposed a malware detection virtualization-based approach that converts the binary files of malware into RGB images and then extracts the features from these images using an ensemble method containing two neural networks: Deep-CNN and Deep-GAN. Landman et al. [1] detected unknown malware in a cloud environment based on Linux. Landman et al. [1] built a framework called Deep-Hook that extracts the memory dump sequences of a virtual machine while it is working and then reforms them into visual images. Finally, the CNN-based classifier is used to detect the malware based on the visual image. Kumar et al. [5] proposed a DTMIC method for malware classification based on deep transfer learning i.e., a pre-trained CNN model on the ImageNet dataset. DTMIC converts the portable executable files into grayscale images then, each image is processed by pre-trained deep CNN to classify to which malware family belongs. Vasan et al. [12] proposed an IMCFN approach that uses fine-tuning with the CNN model to classify image-based malware. IMCFN converts the binary sequences of malware into RGB images and then, processed them by a fine-tuned CNN model that is trained previously with the ImageNet dataset. Tian et al. [11] proposed MDCHD, a visualization-based malware detection in virtual machine environments. MDCHD collects the control flow of the target software at the run-time using the IPT technique and then converts them into RGB images to be processed by CNN architecture that detects the malware. Makandar et al. [27] use the image processing concepts for malware classification based on SVM multi-class. Using the discrete wavelet transform GIST, Gabor wavelet, and other features, the multi-resolution wavelets are utilized to construct an efficient texture feature vector. Narayanan et al. [28] visualize the viruses and malware in an image. Then, the features are extracted from the images using principal component analysis (PCA), and hybrid techniques are used such as ANN along with SVM and KNN to classify the malware.

Convolutional neural networks (CNNs) are dynamic technology that is mostly used in related studies to detect malware. CNNs have a limited ability to distinguish minute

variations among malware types. They might therefore be unable to identify minute alterations in malicious code that might have an impact on how the malware behaves [12]. Recently, most of the studies applied attention mechanisms along with the CNN model to enhance the performance of CNN. Xu et al. [29] proposed Malbert, a malware detection approach that depends on a pre-trained deep learning model using 12 transformers, each transformer contains an attention layer followed by a deep neural network. Zhang et al. [30] proposed a static ransomware detection technique that uses N-gram opcodes and self-attention-based CNN. In 2021, Dosovitskiy et al. [13] proposed a vision transformer (ViT) approach which is an attention-based approach, to be used as an image-based classifier. ViT was applied in several fields and achieved high performance in all studies. Xu et al. [31] created a multi task classification framework using ViT that is capable of simultaneously predicting four glioma molecular expressions based on MR images. Okolo et al. [16] enhanced the architecture of ViT that can perform better for classifying chest X-ray images. Haurum et al. [32] proposed a multi-scale hybrid ViT to classify sewer defects. Sheynin et al. [33] proposed a method that provides simultaneous coarse-grained local interactions and global interactions.

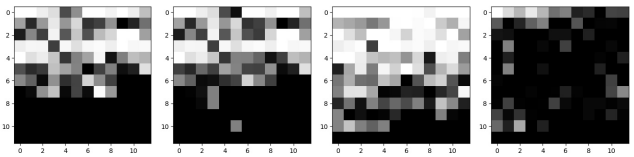
For malware classification and detection, CNNs may not be able to detect small changes in malicious code that could affect how the malware behaves. ViT is a better solution for capturing the spatial information of malware images. Park et al. [34], proposed an enhanced ViT for malware classification. This method incorporates multiple patch encodings which capture both the location information of local features and global features. Seneviratne et al. [35] introduce SHERLOCK, a novel approach for Android malware detection. The method utilizes a Self-Supervised ViT, which is trained on a large corpus of Android application (APK) files. SHERLOCK captures the representations of APK files and based on that, decides whether a given APK file contains malware or not. In this paper, we proposed a B_ViT architecture, a butterfly construction-based vision transformer for malware image classification. The proposed architecture B_ViT tackles the limitations of ViT architecture and image malware-based studies. The proposed malware classifier captures the local spatial representation and global spatial representation of malware images. It does not require domain experts. It supports parallel processing for malware images. Moreover, it is scalable, time-effective, and data-satiable.

III. METHODOLOGY

In this paper, a visualization-based malware classifier/detector namely, butterfly construction-based vision transformer (B_ViT) is proposed. The input of B_ViT model could be any grayscale malware image or a portable executable import after being converted into a grayscale image. As output, B_ViT model classifies the malware image or detects the malware in the input image.

TABLE 1. Notation descriptions used in this paper.

Notation	Descriptions
$x \in R^{H \times W \times C}$	Malware image
$x^{(i)}$	i-th patch of malware image
$P^{(i)}$	i-th flattened patch
$P_E^{(i)}$	i-th patch embedding
k	Parallelism degree
l	Transformers' layer
LTE_{LH}	Local transformer encoder with LH headers
GTE_{GH}	Global transformer encoder with GH headers
$Lpos_E$	Local positional embeddings
$Gpos_E$	Global positional embeddings
Z_i	Local representation of x
$\tilde{Z}_i$	Local transformer embeddings
$\tilde{Z}$	Global representation of x
$\tilde{\tilde{Z}}$	Global transformer embeddings
P_{class}	Probabilities of belonging the image to classes.



(a) Malware (b) Malware (c) Benign (d) Benign

FIGURE 1. Malware and benign samples converted from PE imports.

Prior to delving into the methodology sub-sections, a table of notations 1 will be provided for reference and clarification purposes.

A. GRAYSCALE IMAGE GENERATION PHASE

In this phase, the binary sequence of portable executable (PE) import is converted into a 2-dimensional grayscale image. Then, the image would be used by the proposed model B_ViT to classify or detect the malware. The steps of the PE imports conversion into grayscale images are shown in Algorithm 1. Firstly, the binary sequence of PE import $\{0, 1\}^M$; M is the number of bits, is split into s octets. Each octet of portable executable import $PE[i * 8 + 1], \ldots .PE[i * 8 + 8]$; i is the index of the octet, represents the pixel value P in the grayscale image x so, s is equal to the number of image pixels. Then, the pixel values P_i are converted from binary to an 8-bit unsigned integer where the new values $P'_i \in [0, 255]$. P'_i are normalized to be in the range [0, 1]. The normalized pixels' values P'_i are organized into a vector a with S values. Finally, the vector a is reshaped to a 2-dimensional grayscale image x with (H x W) dimensions and one channel $C = 1$; $S = H * W$. Figure 1 shows some samples of malware and benign images after applying algorithm 1 on the PE imports that are collected from Cuckoo Sandbox reports.

B. THE PROPOSED BUTTERFLY CONSTRUCTION-BASED VISION TRANSFORMER B_ViT

B_ViT is an enhanced vision transformer (ViT) architecture that relies on the parallel processing of image segments and

Algorithm 1 Conversion Malware Portable Executable Import Into Grayscale Image

1: **Input** : Malware PE import: $\{0,1\}^M$
2: **output** : $x \in R^{H\times W\times C}$.
3: $S \leftarrow M/8$
4: $a \leftarrow []$
5: **For** $i : 0 \rightarrow (s-1)$ **do**
6: $\quad P_i \leftarrow PE[i*8+1], \ldots., PE[i*8+8]$
7: $\quad P'_i \leftarrow uint(P) \quad ; P'_i \in [0, 255]$
8: $\quad P''_i \leftarrow Norm(P'_i) \quad ; P''_i \in [0, 1]$
9: $a \leftarrow [P''_0, P''_1, \ldots., P''_{s-1}]$
10: $[x]_{H\times W} \leftarrow [a]_{S\times 1} \quad ; x \in R^{H\times W}, a \in R^{S\times 1}$

captures local and global spatial representations of malware images. B_ViT utilized local and global positional encoding of malware images. Local and global transformer encoders are also included in B_ViT. The architecture of the proposed malware classifier B_ViT is shown in Figure 2. B_ViT has four phases: image partitioning & patches embeddings; local attention; global attention; and training and malware classification. The details of each phase are shown in the following sections.

1) IMAGE PARTITIONING & PATCHES EMBEDDINGS PHASE

The grid partitioning is followed in the proposed model, the malware image $x \in R^{H\times W\times C}$ is partitioned into N non-overlapping square patches in this phase as shown in equation 1 where (H, W) is the malware image resolution, C is the channels number, $(P\times P)$ is patch resolution and $N = \frac{H\times W}{P^2}$. Then, each patch $x^{(i)}; i = 1, \ldots, N$ is flattened i.e, reshaped into a sequence of feature vectors with size $(P \times P \times C)$. A trainable linear projection is used to map the flattened patches $P^{(i)}$ to D dimensions, which are named patch embeddings $P_E^{(i)}$ as shown in equation 2.

$$x \rightarrow [x^{(1)}, x^{(2)}, \ldots., x^{(N)}] \; ; x^{(i)} \in R^{P\times P\times C} \; and \; i = 0, \ldots, N \tag{1}$$

$$P_E^{(i)} = LP(F(x^{(i)})) = Dense_D(F(x^{(i)})); i = 1, \ldots, N \tag{2}$$

where F is the flatten function and LP is the linear projection function i.e, dense layer with D units. The steps from 3 to 8 in Algorithm 2 show the overall process of image partitioning and patch embedding phase. All these steps could be performed simultaneously because there is no dependency between the image patches.

2) LOCAL ATTENTION PHASE

Unlike the original ViT, B_ViT applies self-attention-based transformers to process the malware image in segments simultaneously, where more focus is placed on local features and local representation. k local transformer encoders LTE are used where k is the parallelism degree i.e, the number of threads. The steps of this phase are shown in lines 12-21 in Algorithm 2. To generate the local representation of malware image Z_i, the patch embeddings P_E are grouped into k segments, and each segment's patch embeddings $[P_E^{(i-1)\times\frac{N}{k}+1}, \ldots., P_E^{(i)\times\frac{N}{k}}]$ are added to the local positional embeddings $[Lpos_E(1), \ldots., Lpos_E(k)]$; $\tilde{Lpos}_E \in R^{(\frac{N}{k})\times D}$ which are used to provide information on where each patch was located in the segment then, Z_i is generated as shown in equation 3. The local positional embeddings of each segment are produced by mapping the local patch position into a vector with size D. A trainable linear projection function i.e., a dense layer is used to map the local patch position to D dimensions. After that, for each segment, the learnable embedding $x_{class}^{(0)}$ is mapped into a vector with D size by using linear projection and added to the positional embedding of zero $Lpos_E(0)$. The outcome is prepended to each segment's Z_i as shown in equation 3.

$$\begin{cases} Z_i = [P_E^{(i-1)\times\frac{N}{k}+1} + Lpos_E(1), \ldots\ldots\ldots, P_E^{(i)\times\frac{N}{k}} \\ \qquad +Lpos_E(\frac{N}{k})] \\ Z_i \leftarrow [x_{class}^{(0)}.Linerar\_projection + Lpos_E(0) \;||\; Z_i] \end{cases} \tag{3}$$

where $\tilde{Lpos}_E \in R^{(\frac{N}{k})\times D}$, $i = 1, \ldots, k$ is the index of the segment which is addressed by the i-th local transformer encoder.

Finally, as shown in line 20 of Algorithm 2, each segment Z_i is fed to the corresponding local transformer encoder with LH heads, which produce $\tilde{Z}_i \in R^{(\frac{N}{k}+1)\times D}$. The structure of the transformer encoder is shown in equation 4.

$$\begin{cases} a = MhSa_h(Norm(Z)) + Z \\ \tilde{Z} = MLP(Norm(a)) + a \end{cases} ; Z \in R^{N\times D} \tag{4}$$

where $MhSa_h$ is a multi-head self-attention function with h heads. MLP is a multi-layer perceptron with two hidden layers: 128, 64 units. $\tilde{Z}$ is the N transformer embeddings, each of which has a D size.

3) GLOBAL ATTENTION PHASE

In this phase, the input image is processed in one block by one global transformer encoder, with more focus on global features and representation. The steps of this phase are shown in lines 22-33. To generate the global representation of malware image $\tilde{Z}$, the local transformer embeddings $\tilde{Z}_i$ are concatenated to form one vector $\{\tilde{Z}_1 \;||\; \tilde{Z}_2, \ldots \;||\; \tilde{Z}_k\}$, these vector embeddings excluding the first member from each $\tilde{Z}_i$ are added to patch embeddings $P_E^{(j)}$ and the global position embeddings $Gpos_E(j)$; $Gpos_E \in R^{N\times D}$ which are used to provide information on where each patch was located in the malware image, then, $\tilde{Z}$ is generated as shown in equation 5. The global positional embeddings of the input image are produced by mapping the global patch position into a vector with size D.

$$\tilde{Z} \leftarrow \{\tilde{Z}_1 \;||\; \tilde{Z}_2, \ldots \;||\; \tilde{Z}_k\} \backslash \{\tilde{Z}_i^{(0)}\} + P_E^{(j)} + Gpos_E(j); \; Gpos_E \in R^{N\times D}; j = 1, \ldots, N; \tilde{Z} \in R^{(N+k)\times D} \tag{5}$$

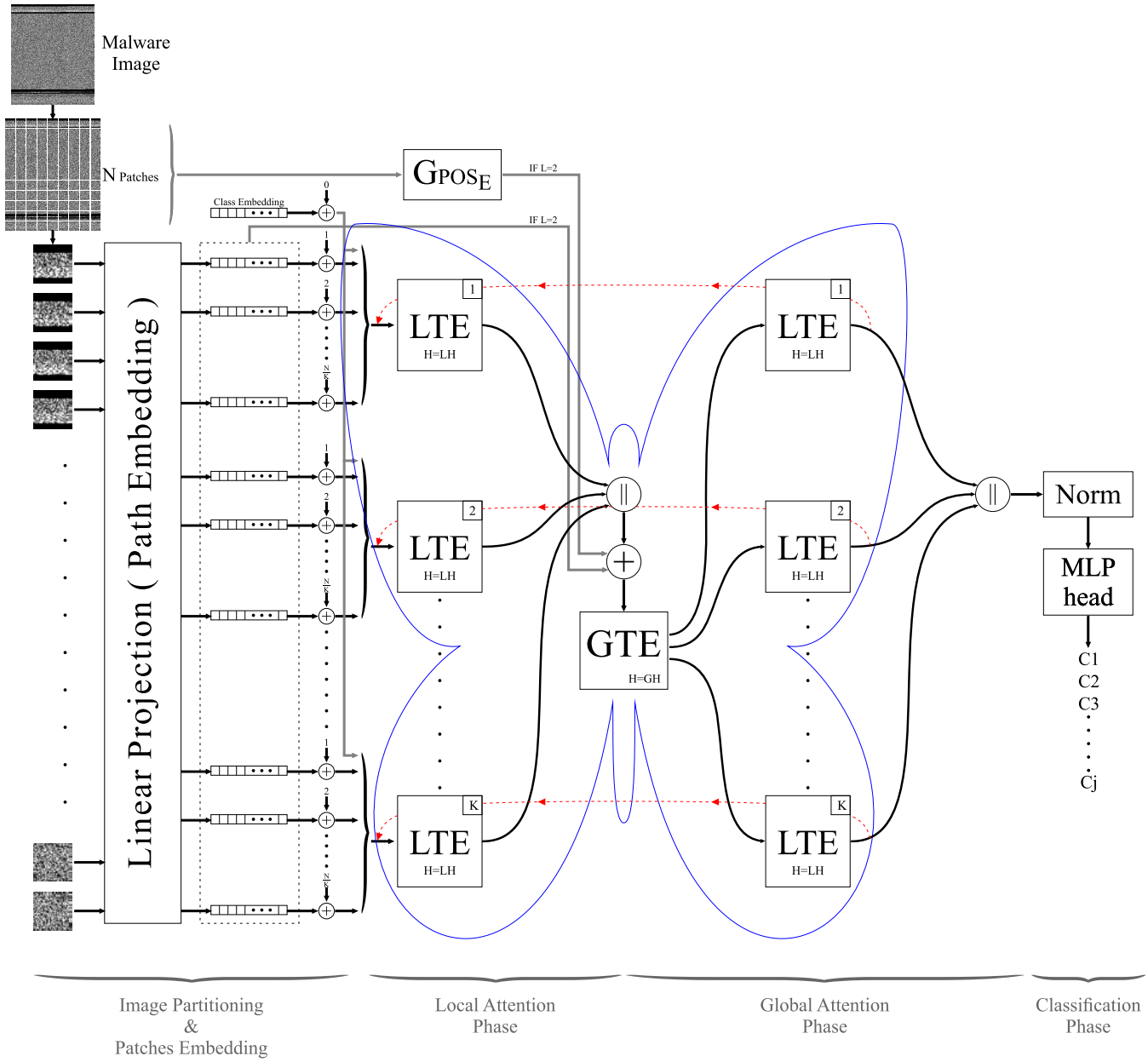


FIGURE 2. Butterfly construction-based vision transformer (B ViT).

where $i = 1, \ldots, k$ is the index of the segment which is addressed by the i-th transformer encoder.

After that, as shown in line 28 of Algorithm 2, $\tilde{Z}$ is fed to the global transformer encoder with GH heads, to produce $\tilde{\tilde{Z}} \in R^{(N+k)\times D}$. The structure of the transformer encoder is shown in equation 4.

Finally, the global transformer embeddings $\tilde{\tilde{Z}}$ are partitioned into k segments, each which $\tilde{\tilde{Z}}_i$ has $\frac{N+K}{k}$ embeddings as shown in equation 6. As shown in line 31 of Algorithm 2, each segment $\tilde{\tilde{Z}}_i$ is fed to the transformer encoder with LH heads to produce $\hat{Z}_i \in R^{(\frac{N}{k}+1)\times D}$. The transformer encoders address the $\tilde{\tilde{Z}}_i$'s simultaneously. The structure of the transformer encoder is shown in equation 4.

$$\tilde{\tilde{Z}}_i \leftarrow [\tilde{\tilde{Z}}_i^{(i-1)\times\frac{N+k}{k}+1}, \ldots\ldots\ldots, \tilde{\tilde{Z}}_i^{(i)\times\frac{N+k}{k}}] \tag{6}$$

4) TRAINING AND MALWARE CLASSIFICATION PHASE

B_ViT model implements same ViT-adopted variants [13] and IEViT-adopted variants [16]. Therefore, four variants of B_ViT such as BViT/B16, BViT/B32, BViT/L16, and BViT/L32 have been experimented. Table 2 shows the details of ViT-adopted, IEViT-adopted, and B_ViT-adopted variants. More details in all the variants of B_ViT such as BViT/B16, BViT/B32, BViT/L16, and BViT/L32 are provided as follows:

Algorithm 2 B_VIT: Butterfly Construction-Based Vision Transformer Algorithm

Input : $[x] \in R^{H\times W\times C}, P, LH, GH, D, MLP\_Size$
output : (B_ViT) optimal classifier.
$N \leftarrow (H * W)/P^2$
$[x^{(1)}, x^{(2)}, \ldots.., x^{(N)}] \leftarrow x \;\; ; x^{(i)} \in R^{P\times P\times C}$
For $i : 1 \rightarrow N$
    $P^{(i)} \leftarrow x^{(i)}.flatten \;\; ; P^{(i)} \in R^{P.P.C\times 1}$
    $P_E^{(i)} \leftarrow P^{(i)}.Linear\_projection \quad ; P_E^{(i)} \in R^{(P.P.C\times D)}$
END For
$Transformer\_No \leftarrow k$
$L \leftarrow 0$
While $B\_VIT \;\; is \;\; not \;\; trained$
    $L = L + 1$
    For $i : 1 \rightarrow Transformer\_No$
        IF $L == 1 \;\; Do$
            $Z_i \leftarrow [P_E^{(i-1)\times \frac{N}{k}+1} + Lpos_E(1), \ldots\ldots\ldots, P_E^{(i)\times \frac{N}{k}} + Lpos_E(\frac{N}{k})] \;\; ; \tilde{Lpos}_E \in R^{(\frac{N}{k})\times D}$
            $Z_i \leftarrow [x_{class}^{(0)}.Linerar\_projection + Lpos_E(0) \;||\; Z_i] \;\; ; Z_i \in R^{(\frac{N}{k}+1)\times D}$
        $ELSE$
            $Z_i \leftarrow \hat{Z}_i$
        $ENDIF$
        $\tilde{Z}_i \leftarrow LTE_{LH}(Z_i) \quad ; \tilde{Z}_i \in R^{(\frac{N}{k}+1)\times D}$
    END For
    $L = L + 1$
    IF $L == 2 \;\; Do$
        $\tilde{Z} \leftarrow \{\tilde{Z}_1 \;||\; \tilde{Z}_2, \ldots \;||\; \tilde{Z}_k\} \backslash \{\tilde{Z}_i^{(0)}\} + P_E^{(j)} + Gpos_E(j); Gpos_E \in R^{N\times D}; j = 1, \ldots, N; \tilde{Z} \in R^{(N+k)\times D}$
    ELSE
        $\tilde{Z} \leftarrow [\tilde{Z}_1 \;||\; \tilde{Z}_2, \ldots \;||\; \tilde{Z}_k] \;\; ; \tilde{Z} \in R^{(N+k)\times D}$
    $ENDIF$
    $\tilde{\tilde{Z}} \leftarrow GTE_{GH}(\tilde{Z}) \quad ; \tilde{\tilde{Z}} \in R^{(N+k)\times D}$
    $L = L + 1$
    For $i : 1 \rightarrow Transformer\_No$
        $\tilde{\tilde{Z}}_i \leftarrow [\tilde{\tilde{Z}}_i^{(i-1)\times \frac{N+k}{k}+1}, \ldots\ldots\ldots, \tilde{\tilde{Z}}_i^{(i)\times \frac{N+k}{k}}]$
        $\hat{Z}_i \leftarrow LTE_{LH}(\tilde{\tilde{Z}}_i) \quad ; \hat{Z}_i \in R^{(\frac{N}{k}+1)\times D}$
    END For
END While

TABLE 2. A description of ViT, IEViT, and B_ViT variants.

Model	ViT/B16	ViT/B32	ViT/L16	ViT/L32	IEViT/B16	IEViT/B32	IEViT/L16	IEViT/L32	BViT/B16	BViT/B32	BViT/L16	BViT/L32
Patch size (P x P)	16 × 16	32 × 32	16 × 16	32 × 32	16 × 16	32 × 32	16 × 16	32 × 32	16 × 16	32 × 32	16 × 16	32 × 32
Hidden size (D)	768	768	1024	1024	768	768	1024	1024	768	768	1024	1024
MLP size	3072	3072	4096	4096	3072	3072	4096	4096	3072	3072	4096	4096
Global Heads (GH)	12	12	16	16	12	12	16	16	12	12	16	16
Local Heads (LH)	-	-	-	-	-	-	-	-	4	4	4	4
Layers (l)	12	12	24	24	12	12	24	24	12 (4 rounds)	12 (4 rounds)	24 (8 rounds)	24 (8 rounds)
No of Params	85.8M	87.5M	303.3M	305.5M	90.8M	92.4M	309.9M	312.1M	174.2M	177.4M	425.6M	428.3M

- BViT/B16 is a medium-sized Vision Transformer with a patch size of 16 × 16 pixels. It is suitable for tasks with fewer classes or limited computational resources. Each 16 × 16 patch is linearly projected and processed by 12 Transformer encoder layers executed in 4 rounds. The local transformer encoders have 4 headers and the global transformer encoders have 12 headers. These transformer encoders generate embeddings with a size of 768.
- BViT/B32 is a medium-sized Vision Transformer with a patch size of 32 × 32 pixels. It captures a coarser-grained representation by reducing the number of patches compared to BViT/B16. It is suitable for lower-resolution images or when spatially localized features are more prevalent. Each 32 × 32 patch is linearly projected and processed by 12 Transformer encoder layers executed in 4 rounds. The local transformer encoders have 4 headers and the global transformer encoders have 12 headers.

These transformer encoders generate embeddings with a size of 768.

- BViT/L16 is a larger Vision Transformer with a patch size of 16 × 16 pixels. It has a higher capacity for capturing fine-grained details and complex patterns in images. It is suitable for tasks with many classes or when higher accuracy is prioritized over computational efficiency. Each 16 × 16 patch is linearly projected and processed by 24 Transformer encoder layers executed in 8 rounds. The local transformer encoders have 4 headers and the global transformer encoders have 16 headers. These transformer encoders generate embeddings with a size of 1024.
- BViT/L32 is a larger Vision Transformer with a patch size of 32 × 32 pixels. It captures a more holistic view of the input image, considering broader context and global dependencies. It is advantageous for higher-resolution images or those with large-scale spatial structures. Each 32 × 32 patch is linearly projected and processed by 24 Transformer encoder layers executed in 8 rounds. The local transformer encoders have 4 headers and the global transformer encoders have 16 headers. These transformer encoders generate embeddings with a size of 1024.

For training all B_ViT variants, the local attention and global attention phases are iterated several rounds to fine-tune the training parameters of transformer encoders. In each round, three transformers' layers are performed where $l = 1, 2, 3$ are performed in round 1; $l = 4, 5, 6.$ are performed in round 2, and so on. The local attention phase has one transformer encoders layer, in which either the local representation of the malware image Z_i is addressed in the first round i.e, $l = 1$, or the output of transformer encoders in the global attention phase $\hat{Z}_i$ is addressed for the upcoming rounds i.e, $l = 4, 7, 10, \ldots$ as shown in line 18 of Algorithm 2. The global attention phase has two transformer encoders layers, in the first one either the global representation of the malware image along with the output of local transformer encoders $\{\tilde{Z}_1 \,||\, \tilde{Z}_2, \ldots \,||\, \tilde{Z}_k\} \backslash \{\tilde{Z}_i^{(0)}\} + P_E^{(j)} + Gpos_E(j)$ is addressed in the first round i.e, $l = 2$ or, the only output of local transformer encoders $\{\tilde{Z}_1 \,||\, \tilde{Z}_2, \ldots \,||\, \tilde{Z}_k\}$ is addressed for the upcoming rounds i.e,$\ $ $l = 5, 8, 11, \ldots$ as shown in line 26 of Algorithm 2. In the second one, the output of the global transformer encoder $\tilde{\tilde{Z}}$ is addressed for all rounds i.e., $l = 3, 6, 9, 12, \ldots$.

B_ViT is a transfer learning-based model, whereby the pre-trained ViT model on the ImageNet dataset [36] ($\geq$ 10 million images) is used to transfer and initialize the training parameters of the transformers. B_ViT i.e, the target model is then fine-tuned to better fit the malware classification task. This can be done by adding MLP head classifier to the proposed model. In context, B_ViT is time-effective (training the model from scratch is not required), and data-satiable (a reasonable size dataset is sufficient for training along with overcoming the overfitting issue). MLP head classifier is trained on three datasets i.e, MalImg [9], Microsoft BIG [18], and top-1000 PE imports. Two dense layers: 1024, 512 units are included in MLP head classifier beside, one output layer with M units where M is equal to the number of classes for dataset being trained. Finally, the class of malware image is determined by MLP head based on the output of j transformer encoders $\tilde{\tilde{Z}}_i$ as shown in equation 7.

$$\begin{cases} \hat{Z} = \{\hat{Z}_1 || \hat{Z}_2 || \ldots . || \hat{Z}_k\} \\ P_{class} = MLP_{head}(\hat{Z}) \end{cases} ; \hat{Z} \in R^{N \times D} \tag{7}$$

where P_{class} are the probabilities of belonging the image to classes.

The use of transfer learning in B_ViT, being an efficient model, brings several benefits. Without transfer learning, B_ViT would face the following challenges:

1) Training from scratch: B_ViT would require an extensive amount of labelled malware images (ranging from 14 million to 300 million) and significant computational resources, leading to lengthy training times spanning weeks.
2) Generalization limitations: B_ViT's performance may suffer when encountering new or unknown malware samples since it lacks the knowledge and patterns obtained from pretraining on a large-scale dataset.
3) Overfitting risks: In scenarios with limited training data, B_ViT would be more susceptible to overfitting, compromising its ability to generalize well to unseen malware instances.

The proposed model *B_ViT* is experimented and trained with different learning rates, optimizers, and batch sizes and observed how they affect the model's performance. The best performance of *B_ViT* was achieved by using the following parameters: learning rate = 10^{-4}, optimizer = adam, batch size = 16, the loss function is cross-entropy defined in equation 8.

$$L_{CE} = -\sum_{j=1}^{M} d_j log(y_j) \tag{8}$$

where M is the number of classes, d_j is the target label for the j-th class, y_j is the predicted label for the j-th class.

Data augmentation is applied in deep learning-based image classification approaches during the training phase to increase the number of training samples and overcome the overfitting issue [37]. Data augmentation includes some methods such as shearing, scaling, flipping, shifting, rotating, and zooming [38]. Table 3 shows the augmentation steps that are followed in this paper to generate more training data from the original malware images. Furthermore, the testing images are excluded from the augmentation for getting realistic results from the original images.

As shown in Figure 3, Figure 4, and Table 7, the three datasets that are used for training B_ViT are imbalanced i.e., some malware families images are significantly larger or smaller compared to other malware families. However, by increasing the malware images in some classes via data

TABLE 3. Malware Images Augmentation steps.

Augmentation Step	Value
Random vertical flipping	True
Random horizontal flipping	True
Transposing at random factor	>0.75
Rotation at angle	{0, 90, 180, 275}
Rescaling by factor	1/255
Random Saturation within a range	[0.7 1.3]
Random Contrast within a range	[0.8 1.2]
Random Brightness within a range	[-0.1 0.1]

TABLE 4. Malimg data augmentation details.

Malware Family	Testing Samples	Training Samples (before augmentation)	Training Samples (after augmentation)
Adialer.C	24	98	3000
Agent.FYI	23	93	3000
Allaple.A	589	2360	3000
Allaple.L	318	1273	3000
Alueron.gen!J	39	159	3000
Autorun.K	21	85	3000
C2LOP.gen!g	40	160	3000
C2LOP.P	29	117	3000
Dialplatform.B	35	142	3000
Dontovo.A	32	130	3000
Fakerean	76	305	3000
Instantaccess	86	345	3000
Lolyda.AA1	42	171	3000
Lolyda.AA2	36	148	3000
Lolyda.AA3	24	99	3000
Lolyda.AT	31	128	3000
Malex.gen!J	27	109	3000
Obfuscator.AD	28	114	3000
Rbot!gen	31	127	3000
Skintrim.N	16	64	3000
Swizzor.gen!E	25	103	3000
Swizzor.gen!I	26	106	3000
VB.AT	81	327	3000
Wintrim.BX	19	78	3000
Yuner.A	160	640	3000

TABLE 5. Microsoft BIG data augmentation details.

Malware Family	Testing Samples	Training Samples (before augmentation)	Training Samples (after augmentation)
Lollipop	495	1983	3000
Vundo	95	380	3000
Tracur	150	601	3000
Obfuscator.ACY	245	983	3000
Ramnit	308	1233	3000
Kelihos_ver3	588	2354	3000
Simda	8	34	3000
Kelihos_ver1	79	319	3000
Gatak	202	811	3000

augmentation, the impact of unbalanced data can be limited. The details of data augmentation such as training samples (before augmentation), training samples (after augmentation), and testing samples for Malimg, Microsoft BIG, and PE imports are presented in Table 4, Table 5, and Table 6 respectively.

IV. DATASETS

The variants of the proposed model i.e, BViT/B16, BViT/B32, BViT/L16, and BViT/L32 are trained and evaluated on three publicly malware images datasets i.e, MalImg [9], Microsoft BIG [18], and top-1000 PE imports [39]. Tables 7, 8, and 9 show details of MalImg, Microsoft BIG,

TABLE 6. PE imports data augmentation details.

Class	Testing Samples	Training Samples (before augmentation)	Training Samples (after augmentation)
Malware	9130	36521	50000
Benign	386	1544	50000

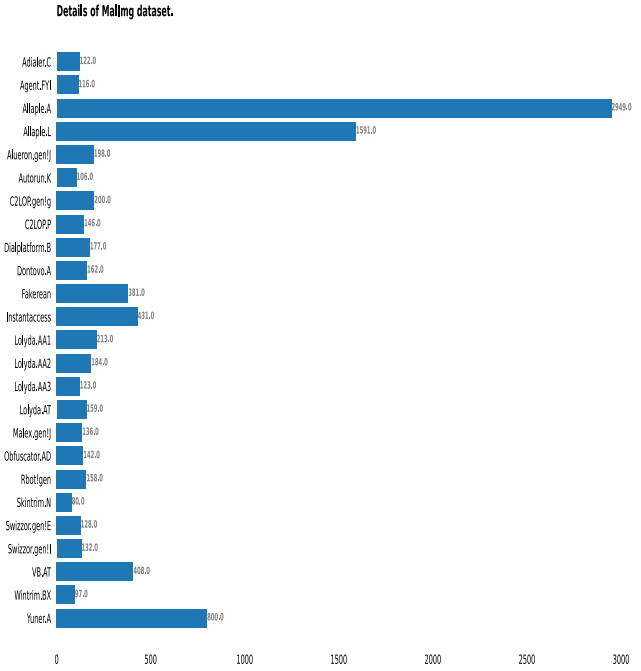


FIGURE 3. Details of MalImg dataset.

and top-1000 PE imports respectively. Malimg contains over 9,000 malware grayscale images distributed in 25 malware families as shown in Figure 3. Microsoft BIG contains over 10,000 malware bytes files represented in hexadecimal and distributed in 9 malware families as shown in Figure 4. Top-1000 imported functions are applicable for malware detection, containing over 47,000 portable executable imports distributed in 2 classes benign, malware as shown in Table 7, and collected from virusshare.com. In experiments, the MalImg and Microsoft BiG, and PE imports are divided into an 80:20 ratio for training and testing.

The implementation of the proposed architecture *B_ViT* was performed using 64-bit windows 10 operating system, Intel Xeon(R) CPU 4 GHz, NVIDIA Geforce GTX 1080 Ti graphics card with CUDA 11.2, 64 GB RAM. Python 3.9.12 (Spyder IDE) with TensorFlow, Keras, and vit_keras libraries are used for the implementation.

V. ANALYSIS OF OBFUSCATION RESISTANCE

Obfuscation is a variety of techniques such s code encryption, code scrambling, and code packing used by malware authors to make it evade detection and remain active on a compromised system for a longer time. Polymorphic and metamorphic obfuscations are two common circumvention techniques [40].

TABLE 7. Details of PE imports.

	No of malware samples	No of benign samples
PE imports	45651	1930

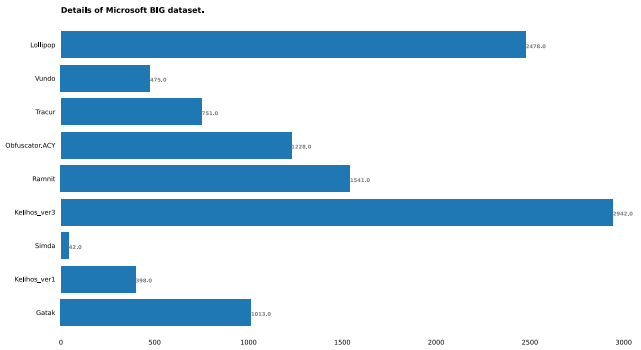


FIGURE 4. Details of Microsoft BIG dataset.

Polymorphic obfuscation alters the appearance and signature of malware using an encryption key. It uses a self-replicating code and mutation engine to swiftly morph malware's code and continuously modify its shape by changing the encryption keys, many signatures are generated subsequently, it is a big challenge to prevent and detect malware using signatures-based detection methods.

Metamorphic obfuscation rewrites the code of the malware itself to generate different malware at each iteration. The new malware's generated without using an encryption key and the functionality of them keeps same. Four methods for metamorphic obfuscation are frequently used: instruction substitution, register reassignment, dead-code insertion, and code transposition.

Over than 80% of malware binary sequences employ obfuscation techniques to evade detection. Obfuscation techniques can be overcome by texture-based malware classification [12], [41]. The texture-based features of malware images are obtained by converting the binary sequence into a 2-dimensional grayscale image as shown in algorithm 1. Then, B_ViT classifies the malware image based on texture features. The proposed study considers the malware families in three datasets i.e, MalImg, Microsoft BIG, and top-1000 PE imports as well as focuses only on polymorphic obfuscation. The results show the efficiency of texture-based malware detection as well as the resilience of B_ViT to polymorphic obfuscation. Some obfuscated malware samples from the ObfuscatorAD family in malImg as shown in Figure 5.

VI. PERFORMANCE ANALYSIS

The proposed model, B_ViT (butterfly construction-based vision transformer) is experimented and evaluated for visualization-based malware classification and detection. Malware classification involves categorizing malware into classes based on characteristics, behaviour, or attributes.

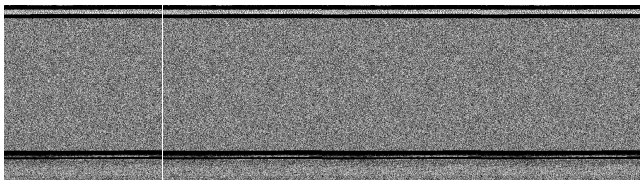


FIGURE 5. Obfuscated malware samples from the ObfuscatorAD family in MalImg.

Malware detection identifies and alerts the presence of malware. However, detection is the main goal of system security but classification aids in developing detection techniques. Malware detection entails a classification process that distinguishes between malware data samples and benign data samples. MalImg [9] and Microsoft BIG [18] datasets are used to evaluate all the variants of B_ViT such as BViT/B16, BViT/B32, BViT/L16, and BViT/L32 for malware classification. Moreover, we conducted an experiment on Microsoft PE imports to detect malware, where the data samples were categorized into two classes: "malware" and "benign". Furthermore, in the three datasets, the B_ViT variants such as BViT/B16, BViT/B32, BViT/L16, and BViT/L32 are compared with the respective variants of IEViT and ViT. The performance of B_ViT, IEViT, and ViT is measured using the most common metrics such as accuracy, recall, precision, and f1-score which are defined in equations 9, 10, 11, and 12 respectively. Moreover, the parallel performance of B_ViT variants is analyzed. Finally, B_ViT is compared with the previous state-of-the-art visualization-based malware classifiers.

$$Accuracy = \frac{TP + TN}{FN + TN + TP + FP} \tag{9}$$

$$Recall = \frac{TP}{FN + TP} \tag{10}$$

$$Precision = \frac{TP}{FP + TP} \tag{11}$$

$$F1 - score = \frac{2 * Recall * Precision}{Recall + Precision} \tag{12}$$

where FP(false-positive) is the number of malware that are classified incorrectly as legitimate software, FN(false-negative) is the number of legitimate software that are classified incorrectly as malware, TP (true-positive) is the number of malware classified correctly, and TN(true-negative) is the number of legitimate software classified correctly.

A. COMPARISON OF B_ViT WITH IEViT AND ViT FOR MalImg DATASET

Comparative analysis of the proposed method B_ViT with IEViT and ViT for image malware classification using MalImg dataset is performed. Figure 6 shows the comparative analysis results of B_ViT variants with IEViT and ViT variants. The accuracy of BViT/B16, BViT/B32, BViT/L16, and BViT/L32 for 25 malware families compared

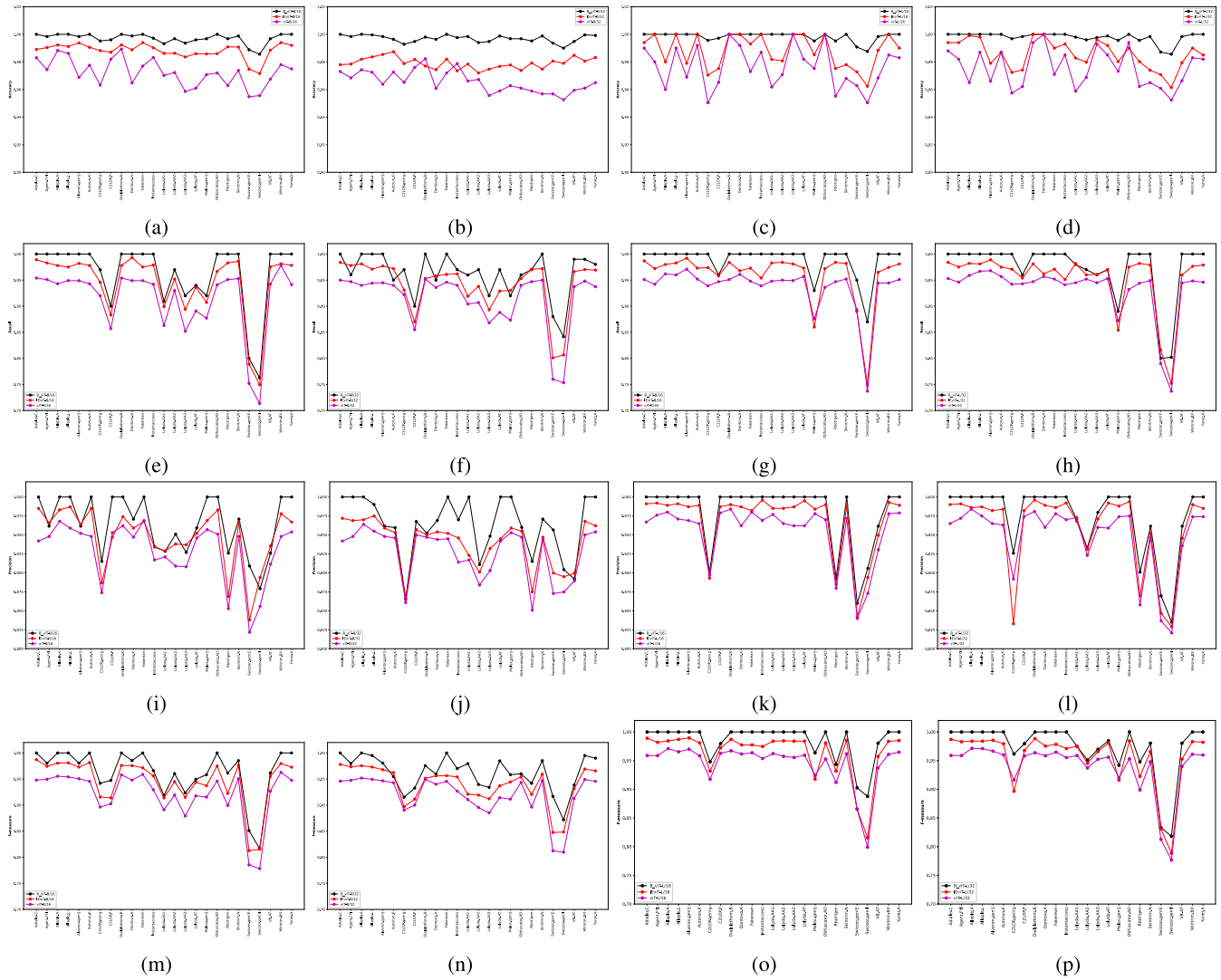


FIGURE 6. comparative analysis of B_ViT variants with IEViT and ViT variants in terms of accuracy, recall, precision, and F1-score for malImg.

to IEViT/B16, IEViT/B32, IEViT/L16, and IEViT/L32 as well as ViT/B16, ViT/B32, ViT/L16, and ViT/L32 is shown in Figure 6 (a, b, c, d) respectively. The recall of BViT/B16, BViT/B32, BViT/L16, and BViT/L32 for 25 malware families compared to IEViT/B16, IEViT/B32, IEViT/L16, and IEViT/L32 as well as ViT/B16, ViT/B32, ViT/L16, and ViT/L32 is shown in Figure 6 (e, f, g, h) respectively. The precision of BViT/B16, BViT/B32, BViT/L16, and BViT/L32 for 25 malware families compared to IEViT/B16, IEViT/B32, IEViT/L16, and IEViT/L32 as well as ViT/B16, ViT/B32, ViT/L16, and ViT/L32 is shown in Figure 6 (i, j, k, l) respectively. The F1-score of BViT/B16, BViT/B32, BViT/L16, and BViT/L32 for 25 malware families compared to IEViT/B16, IEViT/B32, IEViT/L16, and IEViT/L32 as well as ViT/B16, ViT/B32, ViT/L16, and ViT/L32 as shown in Figure 6 (m, n, o, p) respectively. It has been noted that B_ViT variants outperform the IEViT and ViT variants in visualization-based malware classification using MalImg dataset where accuracy, recall, precision, and F1-score are very close to 1 in most malware families.

The confusion matrices of B_ViT variants i.e, BViT/B16, BViT/B32, BViT/L16, and BViT/L32 for MalImg dataset are shown in Figure 7 (a, b, c, d) respectively. The Swizzor.gen!E and Swizzor.gen!I families are difficult to identify from one another in MalImg datasets since they are so similar as shown in Figure 8. The variants of the proposed model BViT/B16, BViT/B32, BViT/L16, and BViT/L32 achieve 98.8%, 99.3%, 99.08%, and 98.72% accuracies for Swizzor.gen!E respectively; and 98.5%, 99.0%, 98.76%, and 98.56% accuracies for Swizzor.gen!I respectively. Therefore, B_ViT is an effective malware classifier even for malware families that are difficult to distinguish.

The overall comparative analysis of malware classification performance of the proposed method B_ViT variants with IEViT and ViT variants for MalImg dataset is shown in Table 8. It has been noted that B_ViT variants outperform

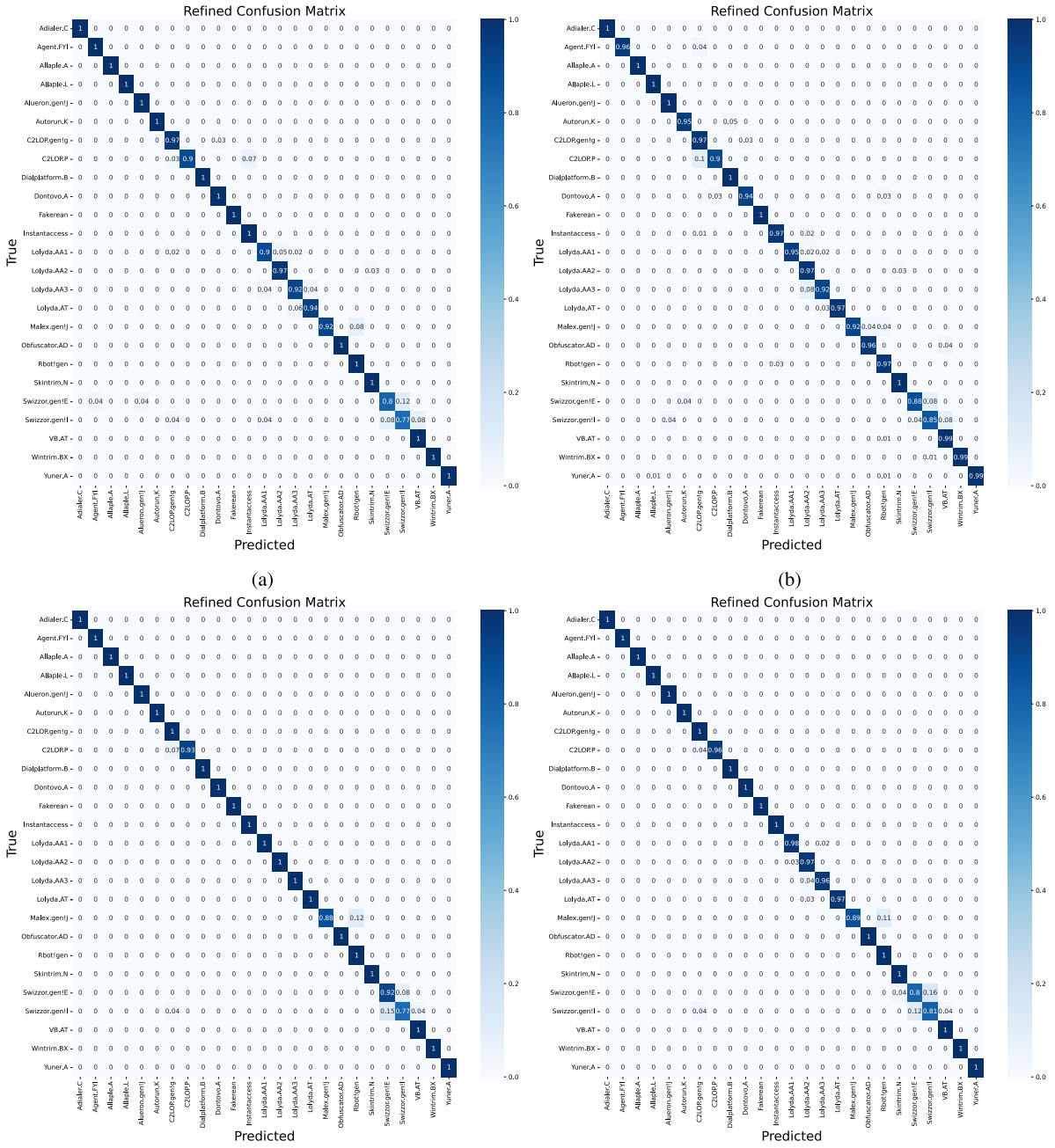


FIGURE 7. The confusion matrices of B_ViT variants: (a) BViT/B16, (b) BViT/B32, (c) BViT/L16, and (d) BViT/L32 for malImg.

the IEViT and ViT variants in visualization-based malware classification using MalImg dataset in terms of accuracy, recall, precision, and F1-score.

B. COMPARISON OF B_ViT WITH IEViT AND ViT FOR MICROSOFT BIG DATASET

Comparative analysis of the proposed method B_ViT with IEViT and ViT for image malware classification using Microsoft BIG dataset is performed. Figure 9 shows the comparative analysis results of B_ViT variants with IEViT and ViT variants. The accuracy of BViT/B16, BViT/B32, BViT/L16, and BViT/L32 for 9 malware families compared to IEViT/B16, IEViT/B32, IEViT/L16, and IEViT/L32 as well as ViT/B16, ViT/B32, ViT/L16, and ViT/L32 is shown in Figure 9 (a, b, c, d) respectively. B_ViT variants consistently achieve higher accuracy compared to IEViT and ViT variants across all malware families, with accuracy scores nearing 1. Among B_ViT variants, BViT/L16, and BViT/L32 have more stable and higher performance, particularly for simda and elihos_ver1 families because the number of global

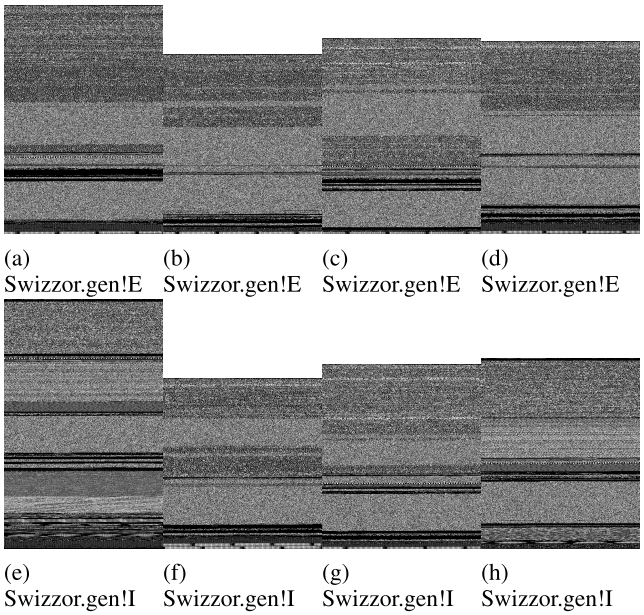


FIGURE 8. Samples of Swizzor.gen!E and Swizzor.gen!I malware famililes.

TABLE 8. Malware classification performance of B_ViT with IEViT and ViT variants using MalImg dataset.

Model	Accuracy	Recall	Precision	F1-score
BViT/B16	98.65%	96.36%	96.48%	96.35%
BViT/B32	98.28%	96.20%	96.37%	96.22%
BViT/L16	99.32%	98.84%	98.08%	98.42%
BViT/L32	99.11%	97.32%	97.34%	97.29%
IEViT/B16	98.81%	94.54%	94.77%	94.59%
IEViT/B32	97.91%	94.06%	93.93%	93.94%
IEViT/L16	98.82%	95.91%	96.96%	96.36%
IEViT/L32	98.56%	95.29%	95.84%	95.49%
ViT/B16	97.24%	91.38%	92.89%	92.05%
ViT/B32	96.56%	91.32%	92.58%	91.87%
ViT/L16	97.64%	93.63%	95.45%	94.49%
ViT/L32	97.55%	93.16%	94.75%	93.91%

heads and the number of layers are higher. The recall of BViT/B16, BViT/B32, BViT/L16, and BViT/L32 for 9 malware families compared to IEViT/B16, IEViT/B32, IEViT/L16, and IEViT/L32 as well as ViT/B16, ViT/B32, ViT/L16, and ViT/L32 is shown in Figure 9 (e, f, g, h) respectively. B_ViT variants consistently achieve higher recall compared to IEViT and ViT variants across all malware families, with recall scores nearing 1. Among B_ViT variants, BViT/L16, and BViT/L32 have more stable and higher performance, particularly for simda and elihos_ver1 families because the number of global heads and the number of layers are higher. The precision of BViT/B16, BViT/B32, BViT/L16, and BViT/L32 for 9 malware families compared to IEViT/B16, IEViT/B32, IEViT/L16, and IEViT/L32 as well as ViT/B16, ViT/B32, ViT/L16, and ViT/L32 is shown in Figure 9 (i, j, k, l) respectively. B_ViT variants consistently achieve higher precision compared to IEViT and ViT variants across all malware families, with precision scores nearing 1. Among B_ViT variants, BViT/L16, and BViT/L32 have more stable and higher performance, particularly for simda and elihos_ver1 families because the number of global heads and the number of layers are higher. The F1-score of BViT/B16, BViT/B32, BViT/L16, and BViT/L32 for 9 malware families compared to IEViT/B16, IEViT/B32, IEViT/L16, and IEViT/L32 as well as ViT/B16, ViT/B32, ViT/L16, and ViT/L32 is shown in Figure 9 (m, n, o, p) respectively. B_ViT variants consistently achieve higher F1-score compared to IEViT and ViT variants across all malware families, with F1-score scores nearing 1. Among B_ViT variants, BViT/L16, and BViT/L32 have more stable and higher performance, particularly for simda and elihos_ver1 families because the number of global heads and the number of layers are higher. It has been noted that B_ViT variants outperform the IEViT and ViT variants in visualization-based malware classification using Microsoft BIG dataset where accuracy, recall, precision, and F1-score are very close to 1 in most malware families.

TABLE 9. Malware classification performance of B_ViT with IEViT and ViT variants using Microsoft BIG dataset.

Model	Accuracy	Recall	Precision	F1-score
BViT/B16	98.80%	97.44%	97.52%	97.44%
BViT/B32	98.62%	97.10%	97.20%	97.10%
BViT/L16	99.49%	99.77%	99.77%	99.77%
BViT/L32	99.26%	99.22%	99.22%	99.22%
IEViT/B16	98.94%	96.02%	95.71%	95.82%
IEViT/B32	98.54%	95.91%	96.16%	95.99%
IEViT/L16	99.26%	98.02%	97.73%	97.87%
IEViT/L32	99.15%	97.52%	97.95%	97.73%
ViT/B16	98.63%	95.32%	94.66%	94.94%
ViT/B32	97.94%	95.22%	94.55%	94.83%
ViT/L16	98.92%	97.29%	96.33%	96.80%
ViT/L32	98.72%	95.63%	96.08%	95.86%

The confusion matrices of B_ViT variants i.e, BViT/B16, BViT/B32, BViT/L16, and BViT/L32 for Microsoft BIG dataset are shown in Figure 10 (a, b, c, d) respectively. The simda and kelihos_ver1 families are difficult to identify from one another in Microsoft BIG datasets since they are so similar. The variants of the proposed model BViT/B16, BViT/B32, BViT/L16, and BViT/L32 achieve 98.10%, 98.43%, 100%, and 99.88% accuracies for simda respectively; and 98.10%, 100%, 98.76%, and 99.77% accuracies for kelihos_ver1 respectively. Therefore, B_ViT in particular, BViT/L16, and BViT/L32 are effective malware classifiers even for malware families that are difficult to distinguish.

The overall comparative analysis of malware classification performance of the proposed method B_ViT variants with IEViT and ViT variants for Microsoft BIG dataset is shown in Table 9. It has been noted that B_ViT variants outperform the IEViT and ViT variants in visualization-based malware classification using Microsoft BIG dataset in terms of accuracy, recall, precision, and F1-score.

C. COMPARISON OF B_ViT WITH IEViT AND ViT FOR TOP-1000 PE IMPORTS DATASET

Comparative analysis of the proposed method B_ViT with IEViT and ViT for image malware detection

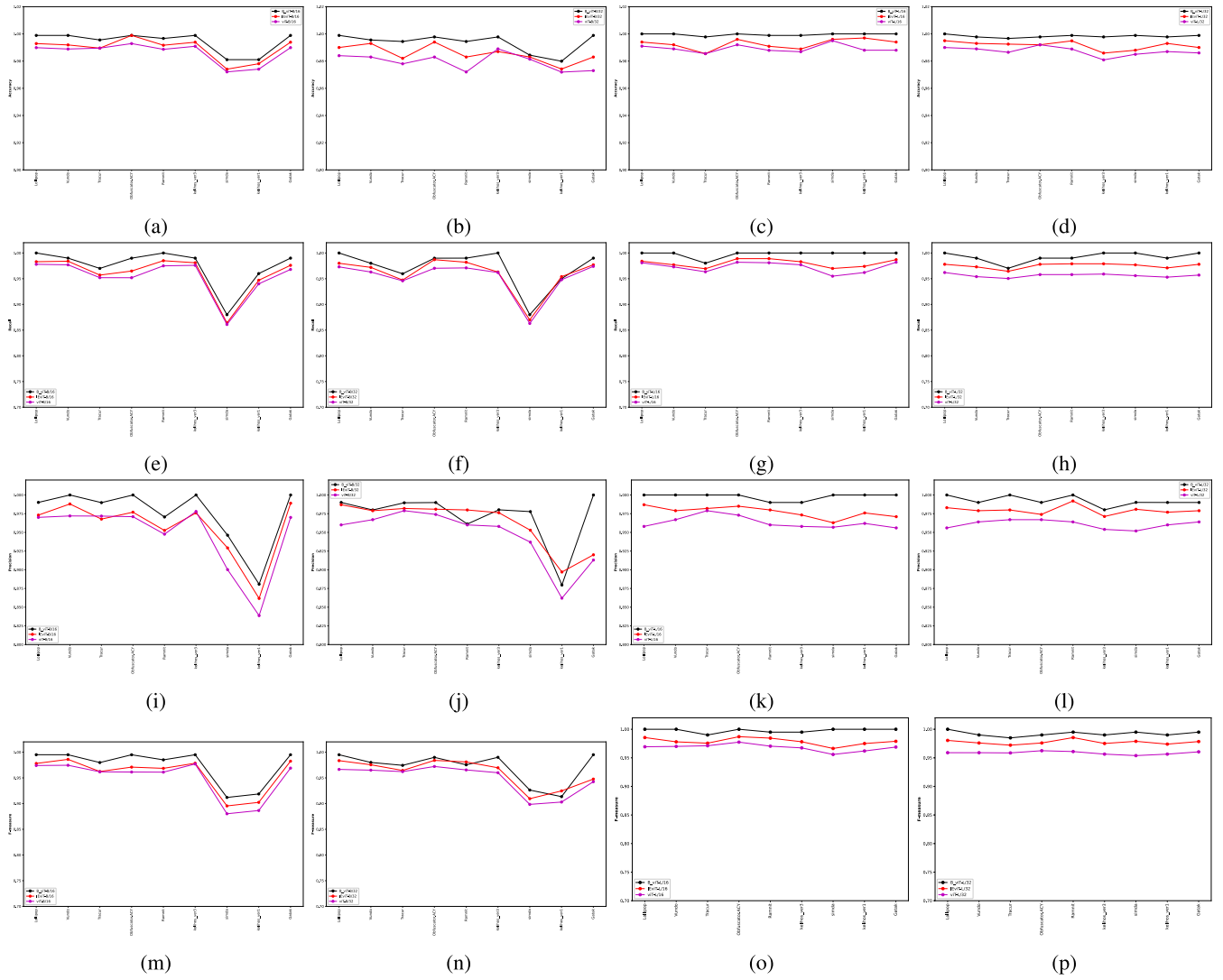


FIGURE 9. comparative analysis of B_ViT variants with IEViT and ViT variants in terms of accuracy, recall, precision, and F1-score for Microsoft BIG dataset.

using the top-1000 PE imports dataset is performed. Figure 11 (a, b, c, d) shows the confusion matrices of B_ViT variants i.e, BViT/B16, BViT/B32, BViT/L16, and BViT/L32 for the top-1000 PE imports dataset. It has been noted that most of the benign and malware samples are correctly classified with a few FP and FN. Therefore, B_ViT is an effective visualization-based malware detector.

The overall comparative analysis of malware detection performance of the proposed method B_ViT variants with IEViT and ViT variants for PE imports dataset is shown in Table 10. It has been noted that B_ViT variants outperform the IEViT and ViT variants in visualization-based malware detection using PE imports dataset in terms of accuracy, recall, precision, and F1-score.

Finally, Tables 11 and 12 show the improvement of B_ViT variants over IEViT and ViT variants in terms of f1-score for MalImg, Microsoft BIG, and PE imports. $\Delta_1, \Delta_2, \Delta_3, \Delta_4$ denote the improvement of B_ViT variants i.e, BViT/B16, BViT/B32, BViT/L16, and BViT/L32 over IEViT variants i.e, IEViT/B16, IEViT/B32, IEViT/L16, and IEViT/L32 respectively. $\Delta_5, \Delta_6, \Delta_7, \Delta_8$ denote the improvement of

TABLE 10. Malware detection performance of B_ViT with IEViT and ViT variants using top-1000 PE imports dataset.

Model	Accuracy	Recall	Precision	F1-score
BViT/B16	99.84%	99.73%	99.85%	99.79%
BViT/B32	99.87%	99.48%	99.36%	99.41%
BViT/L16	99.99%	100%	99.99%	99.99%
BViT/L32	99.97%	99.95%	99.91%	99.92%
IEViT/B16	99.69%	99.53%	99.46%	99.49%
IEViT/B32	99.73%	99.68%	99.70%	99.68%
IEViT/L16	99.76%	99.72%	99.77%	99.74%
IEViT/L32	99.74%	99.73%	99.74%	99.73%
ViT/B16	99.16%	99.03%	99.12%	99.07%
ViT/B32	98.94%	98.91%	98.93%	98.91%
ViT/L16	99.27%	99.26%	99.22%	99.23%
ViT/L32	99.24%	99.24%	99.23%	99.23%

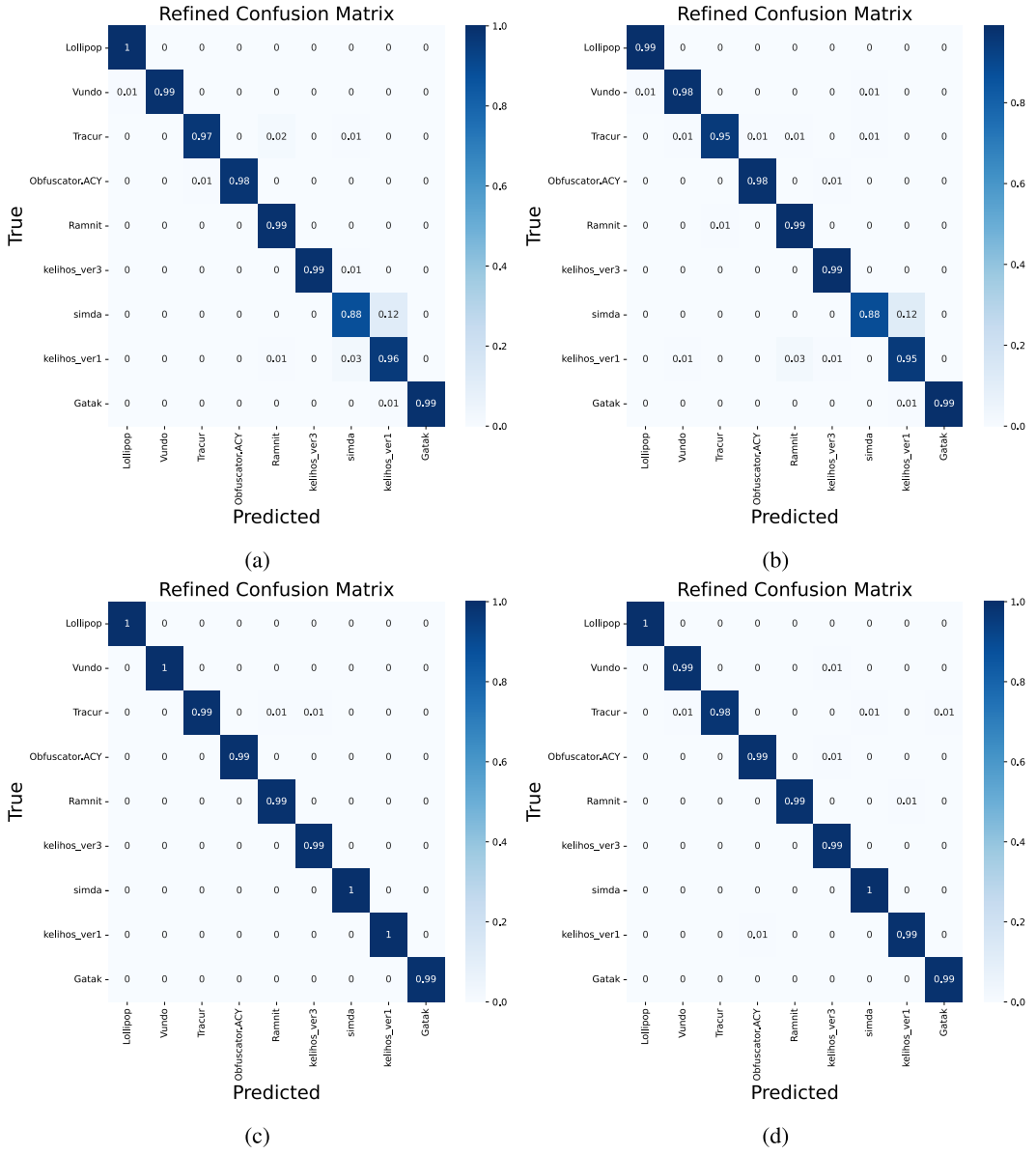


FIGURE 10. The confusion matrices of B_ViT variants i.e, (a) BViT/B16, (b) BViT/B32, (c) BViT/L16, and (d) BViT/L32 for Microsoft BIG dataset.

B_ViT variants i.e, BViT/B16, BViT/B32, BViT/L16, and BViT/L32 over ViT variants i.e, ViT/B16, ViT/B32, ViT/L16, and ViT/L32 respectively. It has been noted that B_ViT variants outperform IEViT and ViT variants. BViT/B16 achieves average improvement in terms of F1-score over IEViT/B16 and ViT/B16 equal to 1.23% and 2.5% respectively. BViT/B32 achieves average improvement in terms of F1-score over IEViT/B32 and ViT/B32 equal to 1.04% and 2.37% respectively. BViT/L16 achieves average improvement in terms of F1-score over IEViT/L16 and ViT/L16 equal to 1.4% and 2.55% respectively. BViT/L32 achieves average improvement in terms of F1-score over IEViT/L32 and ViT/L32 equal to 1.16% and 2.48% respectively.

TABLE 11. Improvement of B_ViT variants over IEViT in terms of f1-score.

Dataset	Δ_1	Δ_2	Δ_3	Δ_4	Avg Δ
MalImg	1.76%	2.28%	2.06%	1.8%	1.98%
Microsoft BIG	1.62%	1.11%	1.9%	1.49%	1.53%
PE imports	0.3%	-0.27%	0.25%	0.19%	0.12%
Average	1.23%	1.04%	1.4%	1.16%	1.21%

Δ_1 : BViT/B16 & IEViT/B16, Δ_2 : BViT/B32 & IEViT/B32, Δ_3 : BViT/L16 & IEViT/L16, Δ_4 : BViT/L32 & IEViT/L32

D. PARALLEL ANALYSIS OF B_ViT AND COMPARISON WITH IEViT AND ViT

B_ViT is a parallel-based architecture that supports the parallel processing of images' patches. Therefore, a parallel

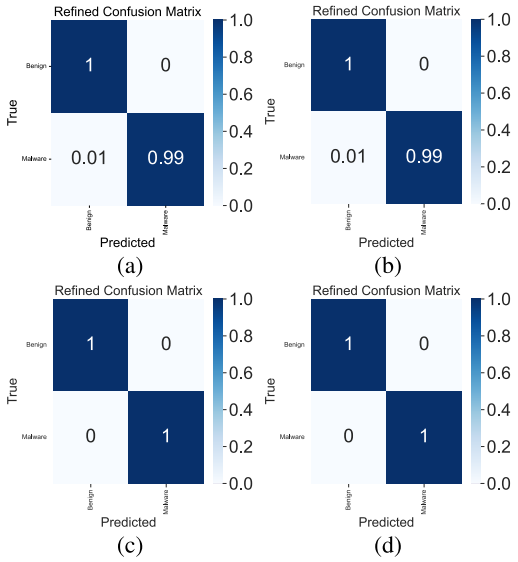


FIGURE 11. The confusion matrices of B_ViT variants i.e, (a) BViT/B16, (b) BViT/B32, (c) BViT/L16, and (d) BViT/L32 for Microsoft PE imports.

TABLE 12. Improvement of B_ViT variants over ViT variants in terms of f1-score.

Dataset	Δ_5	Δ_6	Δ_7	Δ_8	Avg Δ
MalImg	4.3%	4.35%	3.93%	3.38%	3.99%
Microsoft BIG	2.5%	2.27%	2.97%	3.36%	2.78%
PE imports	0.72%	0.5%	0.76%	0.69%	0.67%
Average	2.5%	2.37%	2.55%	2.48%	2.48%

Δ_5 : BViT/B16 & ViT/B16, Δ_6 : BViT/B32 & ViT/B32, Δ_7 : BViT/L16 & ViT/L16, Δ_8 : BViT/L32 & ViT/L32

analysis of B_ViT should be performed and compared with the performance of IEViT and ViT in terms of speed up S, overhead T_0, efficiency E, and running cost C. Speed up S refers to the improvement in performance obtained by using multiple threads in comparison to a single thread. Overhead T_0 refers to the additional time required to manage and coordinate the parallel processing, such as communication and synchronization between processors. Efficiency E refers to the proportion of the total running time that is spent performing useful work, rather than overhead. Running cost C refers to the total time required to run a parallel program on multiple threads. It takes into account both the improvement in performance and the overhead. speed up S, overhead T_0, efficiency E, and running cost C are defined in equations 13, 14, 15, and 16 [42].

$$S = \frac{ET_S(s)}{ET_P(s)} \tag{13}$$

$$T_0 = (k \times ET_P) - ET_S \tag{14}$$

$$E = \frac{S}{k} \tag{15}$$

$$C = k \times ET_P \tag{16}$$

where ET_P is the architecture epoch time running on multiple threads, ET_S is the architecture epoch time running on a single thread, and k is the number of threads.

TABLE 13. Parallel analysis of B_ViT over sequential B_ViT, IEViT, and ViT.

	B_ViT Performance						IEViT Performance		ViT Performance	
variant	ET_S	ET_P	S	T_0	E	C	ET_S	S	ET_S	S
B16	554	146	3.79	30	0.95	584	351	2.40	283	1.94
B32	514	133	3.86	18	0.97	532	227	1.71	164	1.23
L16	985	257	3.83	43	0.96	1028	776	3.02	597	2.32
L32	916	235	3.89	24	0.97	940	603	2.57	411	1.75
Average	742.25	192.75	3.84	28.75	0.96	771	489.25	2.42	363.75	1.81

Table 13 shows the parallel analysis of B_ViT over sequential B_ViT, IEViT, and ViT. It has been noted that an epoch time of B_ViT is smaller than sequential B_ViT; the speed up is approximately equal to k (the number of threads); the overhead is very less compared to running cost; and the efficiency of B_ViT is approximately equal to 1 which is the optimal value. Therefore, the parallel-based architecture of ViT i.e, B_ViT is time-effective for malware classification and detection as well as training. Moreover, the time performance of B_ViT is higher than IEViT and ViT as shown in Table 13. The average speed-up of B_ViT variants over IEViT and ViT variants is equal to 2.42 and 1.81 respectively.

E. PERFORMANCE COMPARISON OF B_ViT WITH RECENT VISUALIZATION-BASED MALWARE CLASSIFIERS

Table 14 presents a comparison between the proposed malware classifier/detector B_ViT and the recent visualization-based malware classifiers in terms of four major performance criteria i.e, accuracy, recall, precision, and F1-score. Moreover, the comparison has been performed with the recent visualization-based malware classifiers that use the same datasets used in this paper. It has been noted that the proposed method that uses B-ViT architecture outperforms recent visualization-based malware classification methods that use CNN architectures. Therefore, it may be concluded that butterfly construction-based vision transformer architecture has a higher performance than CNNs in malware image classification because B_ViT obtains the malware image's local spatial representation and global spatial representation and extracts the features without the need for domain experts and feature engineering. In addition, B_ViT detects the obfuscated malware and the packed malware that is injected into legitimate software. Finally, B_ViT is a scalable model, whereby the pre-trained ViT model on the ImageNet dataset ($\geq$10 million images) is used to transfer and initialize the training parameters of the transformers.

VII. DISCUSSION

Recent studies focus on using CNNs for dynamic visualization-based malware detection, but they struggle to detect subtle differences between malware variants. ViT outperforms CNNs, but few studies explore ViT-based malware classifiers. ViT still has limitations: computationally intensive for large images, lacks local representation, requires abundant training data, and demands powerful hardware and long training times. Therefore, B_ViT

TABLE 14. Comparasion of B_ViT malware classifier/detector with the recent visualization-based malware classifiers.

Methods	Dataset	Accuracy	Recall	Precision	F1-Score
The proposed model (B_ViT)	MalImg	99.32%	98.84%	98.08%	98.42%
The proposed model (B_ViT)	Microsoft BIG	99.49%	99.77%	99.77%	99.77%
The proposed model (B_ViT)	PE imports	99.99%	100%	99.99%	99.99%
Enhanced ViT + Patch Encoding [34]	Microsoft BIG	96.83%	-	-	-
Self supervised ViT [35]	MalNet	97.0%	86.7%	87.8%	99.90%
BHMDC [43]	MalImg	99.90%	99.90%	99.90%	99.90%
BHMDC [43]	Microsoft BIG	99.70%	99.70%	99.70%	99.70%
MCTVD [44]	Microsoft BIG	99.44%	-	-	-
EfficientNetB1 [45]	Microsoft BIG	98%	98%	98%	98%
DTMIC [5]	MalImg	98.92%	99%	99%	99%
IMCFN [12]	MalImg	98.82%	98.81%	98.85%	98.82%
VAE+ CNN (1D) +LSTM [46]	Microsoft BIG	97.47%	-	-	-
NSGA-II [47]	MalImg	97.6%	97.6%	88.4%	92.77%
IDA+DRBA [48]	MalImg	94.50%	94.50%	94.60%	94.55%
VGG16+SVM [49]	Microsoft BIG	99.08%	-	-	-
Cui et. al. [50]	MalImg	97.60%	88.40%	-	-
Bhodia et. al [48]	MalImg	94.80	-	-	-

butterfly construction-based vision transformer model for visualization-based malware classification and detection is proposed. All B_ViT variants i.e, BViT/B16, BViT/B32, BViT/L16, and BViT/L32 are experimented and evaluated using MalImg and Microsoft BIG datasets for image malware classification and compared with the respective variants of IEViT and ViT. The performance comparison demonstrates the superiority of B_ViT architecture over recent visualization-based malware classification methods utilizing both CNN and ViT architectures, as indicated in Table 14 because B_ViT captures the local spatial representation and global spatial representation of malware images. Moreover, B_ViT variants demonstrate enhanced speed (S) improvement, reduced overhead (T0), improved efficiency (E), and optimized running cost (C) compared to IEViT and ViT variants as indicated in Table 11, Table 12, and Table 13. This is achieved through B_ViT's parallel-based architecture, which enables efficient parallel processing of image patches. The comparative analysis of the proposed method B_ViT with IEViT and ViT for image malware classification and detection. B_ViT variants consistently achieve higher accuracy, recall, precision, and F1-score compared to IEViT and ViT variants across all malware families, with values scores nearing 1 as shown in Figure 6 and 9. Among B_ViT variants, BViT/L16, and BViT/L32 have more stable and higher performance, particularly for simda and elihos_ver1 families in Microsoft BIG as shown in Figure 9, Swizzor.gen!E and Swizzor.gen!I families in MalImg as shown in Figure 6. This is achieved through the number of global heads and the number of layers is higher. In Figure 7, it can be observed that 12% of the tested samples originally classified as Swizzor.gen!I family were misclassified as Swizzor.gen!E family, while 16% of the tested samples belonging to Swizzor.gen!E family were misclassified as Swizzor.gen!I family. This misclassification can be attributed to the high similarity between these two families, leading to challenges in distinguishing them accurately. In Figure 10, it can be observed that 12% of the tested samples originally classified as kelihos_ver1 family were misclassified as simda family, while 0% of the tested samples belonging to simda family were misclassified as kelihos_ver1 family. This misclassification can be attributed to the high similarity between these two families, leading to challenges in distinguishing them accurately.

VIII. CONCLUSION

This paper proposes a butterfly construction-based vision transformer (B_ViT) model for visualization-based malware classification and detection. B_ViT is trained and evaluated on grayscale malware images collected from MalImg or Microsoft BIG datasets or converted from portable executable imports. B_ViT has four phases: image partitioning and patches embeddings; local attention; global attention; and training and malware classification. In local attention phase, self-attention-based local transformer encoders along with local positional encoding process the input image's patches simultaneously to capture the local representation and features of malware image. In global attention phase, one self-attention-based global transformer encoder along with global positional encoding process the input image as one block, to capture the global representation and features of malware image. B_ViT is a transfer learning-based model that uses a pre-trained ViT model on the ImageNet dataset to initialize the training parameters of transformers, then the B_ViT is fine-tuned to fit malware classification task. All B_ViT variants i.e, BViT/B16, BViT/B32, BViT/L16, and BViT/L32 are experimented and evaluated using MalImg and Microsoft BIG datasets for image malware classification and compared with the respective variants of IEViT and ViT. The comparative analysis shows that B_ViT variants outperform the IEViT and ViT variants in visualization-based malware classification achieving accuracy equal to 98.65%, 98.28%, 99.32%, and 99.11% in MalImg; and 98.80%, 98.62%, 99.49%, and 99.26% in Microsoft BIG for BViT/B16, BViT/B32, BViT/L16, and BViT/L32 respectively. Besides, B_ViT variants are evaluated using portable executable imports for image malware detection and compared with the respective variants of IEViT and ViT. The comparative analysis shows that B_ViT variants outperform the IEViT and ViT variants in visualization-based malware detection achieving accuracy equal to 99.84%, 99.87%, 99.99%, and 99.97% for BViT/B16, BViT/B32, BViT/L16, and BViT/L32 respectively. The results show that B_ViT achieves average improvement in terms of F1-score over IEViT and ViT equal to 1.21%, and 2.48% respectively. Since B_ViT is a parallel-based architecture, a parallel analysis of B_ViT over sequential B_ViT, IEViT, and ViT is performed. The results show that B_ViT is time-effective for malware classification and detection where the average speed-up of B_ViT variants over sequential B_ViT, IEViT and ViT variants are equal to 3.84, 2.42 and 1.81 respectively. Moreover,

the analysis shows the efficiency of texture-based malware detection as well as the resilience of B_ViT to polymorphic obfuscation. The proposed malware classifier/detector is visualization-based so, does not require domain experts for feature extraction, feature engineering, etc. Finally, the proposed method that uses B_ViT architecture outperforms recent visualization-based malware classification methods that use CNN architectures as well as ViT-based malware classifiers. The utilization of the B_ViT-based malware classifier/detector in practice presents certain limitations that should be acknowledged. Firstly, its implementation necessitates a high-resource platform, especially when employing a high degree of parallelism because more local transformer encoders should be run to capture the local representation of malware images. Secondly, to ensure its effectiveness, the proposed method must be thoroughly tested in an on-site environment. To overcome these limitations, future work should focus on further optimization and refinement of the approach.

REFERENCES

[1] T. Landman and N. Nissim, "Deep-hook: A trusted deep learning-based framework for unknown malware detection and classification in Linux cloud environments," *Neural Netw.*, vol. 144, Dec. 2021, pp. 648–685, doi: 10.1016/j.neunet.2021.09.019.

[2] (2020). *Symantec. Symantec Threat Landscape Report*. [Online]. Available: https://symantecenterprise-blogs.security.com/blogs/threat-intelligence/threat-landscape-q1-2020

[3] J. Suaboot, Z. Tari, A. Mahmood, A. Y. Zomaya, and W. Li, "Sub-curve HMM: A malware detection approach based on partial analysis of API call sequences," *Comput. Secur.*, vol. 92, May 2020, Art. no. 101773, doi: 10.1016/j.cose.2020.101773.

[4] "Major ransomware campaign targets healthcare facilities in US," *Comput. Fraud Secur.*, vol. 2020, no. 11, pp. 1–3, 2020, doi: 10.1016/S1361-3723(20)30112-3.

[5] S. Kumar and B. Janet, "DTMIC: Deep transfer learning for malware image classification," *J. Inf. Secur. Appl.*, vol. 64, Feb. 2022, Art. no. 103063, doi: 10.1016/j.jisa.2021.103063.

[6] A. Bensaoud and J. Kalita, "Deep multi-task learning for malware image classification," *J. Inf. Secur. Appl.*, vol. 64, Feb. 2022, Art. no. 103057, doi: 10.1016/j.jisa.2021.103057.

[7] A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," *Human-centric Comput. Inf. Sci.*, vol. 8, no. 1, pp. 1–22, Dec. 2018.

[8] M. M. Belal and D. M. Sundaram, "Comprehensive review on intelligent security defences in cloud: Taxonomy, security issues, ML/DL techniques, challenges and future trends," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 10, pp. 9102–9131, Nov. 2022, doi: 10.1016/j.jksuci.2022.08.035.

[9] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification," in *Proc. 8th Int. Symp. Visualizat. Cyber Secur.*, Jul. 2011, pp. 1–7.

[10] B. Zou, C. Cao, F. Tao, and L. Wang, "IMCLNet: A lightweight deep neural network for image-based malware classification," *J. Inf. Secur. Appl.*, vol. 70, Nov. 2022, Art. no. 103313, doi: 10.1016/j.jisa.2022.103313.

[11] D. Tian, Q. Ying, X. Jia, R. Ma, C. Hu, and W. Liu, "MDCHD: A novel malware detection method in cloud using hardware trace and deep learning," *Comput. Netw.*, vol. 198, Oct. 2021, Art. no. 108394, doi: 10.1016/j.comnet.2021.108394.

[12] 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, Apr. 2020, Art. no. 107138, doi: 10.1016/j.comnet.2020.107138.

[13] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16×16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.

[14] Y. Borhani, J. Khoramdel, and E. Najafi, "A deep learning based approach for automated plant disease classification using vision transformer," *Sci. Rep.*, vol. 12, no. 1, Jul. 2022, Art. no. 11554.

[15] A. Tabbakh and S. S. Barpanda, "A deep features extraction model based on the transfer learning model and vision transformer 'TLMVi' for plant disease classification," *IEEE Access*, vol. 11, pp. 45377–45392, 2023.

[16] G. I. Okolo, S. Katsigiannis, and N. Ramzan, "IEViT: An enhanced vision transformer architecture for chest X-ray image classification," *Comput. Methods Programs Biomed.*, vol. 226, Nov. 2022, Art. no. 107141, doi: 10.1016/j.cmpb.2022.107141.

[17] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy, "Do vision transformers see like convolutional neural networks?" in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 12116–12128.

[18] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, "Microsoft malware classification challenge," 2018, *arXiv:1802.10135*.

[19] N. Naik, P. Jenkins, N. Savage, L. Yang, T. Boongoen, and N. Iam-On, "Fuzzy-import hashing: A static analysis technique for malware detection," *Forensic Sci. Int., Digit. Invest.*, vol. 37, Jun. 2021, Art. no. 301139, doi: 10.1016/j.fsidi.2021.301139.

[20] C. Liangboonprakong and O. Sornil, "Classification of malware families based on N-grams sequential pattern features," in *Proc. IEEE 8th Conf. Ind. Electron. Appl. (ICIEA)*, Jun. 2013, pp. 777–782, doi: 10.1109/ICIEA.2013.6566472.

[21] M. Narouei, M. Ahmadi, G. Giacinto, H. Takabi, and A. Sami, "DLLMiner: Structural mining for malware detection," *Secur. Commun. Netw.*, vol. 8, no. 18, pp. 3311–3322, 2015.

[22] V. Avdiienko, K. Kuznetsov, A. Gorla, A. Zeller, S. Arzt, S. Rasthofer, and E. Bodden, "Mining apps for abnormal usage of sensitive data," in *Proc. IEEE/ACM 37th IEEE Int. Conf. Softw. Eng.*, vol. 1, May 2015, pp. 426–436.

[23] C. Li, Z. Cheng, H. Zhu, L. Wang, Q. Lv, Y. Wang, N. Li, and D. Sun, "DMALNet: Dynamic malware analysis based on api feature engineering and graph learning," *Comput. Secur.*, vol. 122, Nov. 2022, Art. no. 102872, doi: 10.1016/j.cose.2022.102872.

[24] C. Li, Q. Lv, N. Li, Y. Wang, D. Sun, and Y. Qiao, "A novel deep framework for dynamic malware detection based on API sequence intrinsic features," *Comput. Secur.*, vol. 116, May 2022, Art. no. 102686, doi: 10.1016/j.cose.2022.102686.

[25] L. Chen, S. Sultana, and R. Sahita, "HeNet: A deep learning approach on Intel® processor trace for effective exploit detection," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2018, pp. 109–115.

[26] O. J. Falana, A. S. Sodiya, S. A. Onashoga, and B. S. Badmus, "Mal-detect: An intelligent visualization approach for malware detection," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 5, pp. 1968–1983, May 2022, doi: 10.1016/j.jksuci.2022.02.026.

[27] A. Makandar and A. Patrot, "Malware class recognition using image processing techniques," in *Proc. Int. Conf. Data Manage., Analytics Innov. (ICDMAI)*, Feb. 2017, pp. 76–80, doi: 10.1109/ICDMAI.2017.8073489.

[28] B. N. Narayanan, O. Djaneye-Boundjou, and T. M. Kebede, "Performance analysis of machine learning and pattern recognition algorithms for malware classification," in *Proc. IEEE Nat. Aerosp. Electron. Conf. (NAECON) Ohio Innov. Summit (OIS)*, Jul. 2016, pp. 338–342, doi: 10.1109/NAECON.2016.7856826.

[29] Z. Xu, X. Fang, and G. Yang, "Malbert: A novel pre-training method for malware detection," *Comput. Secur.*, vol. 111, Dec. 2021, Art. no. 102458, doi: 10.1016/j.cose.2021.102458.

[30] B. Zhang, W. Xiao, X. Xiao, A. K. Sangaiah, W. Zhang, and J. Zhang, "Ransomware classification using patch-based CNN and self-attention network on embedded N-grams of opcodes," *Future Gener. Comput. Syst.*, vol. 110, pp. 708–720, Sep. 2020, doi: 10.1016/j.future.2019.09.025.

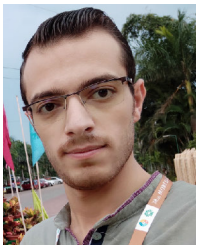
[31] Q. Xu, Q. Q. Xu, N. Shi, L. N. Dong, H. Zhu, and K. Xu, "A multitask classification framework based on vision transformer for predicting molecular expressions of glioma," *Eur. J. Radiol.*, vol. 157, Dec. 2022, Art. no. 110560, doi: 10.1016/j.ejrad.2022.110560.

[32] J. B. Haurum, M. Madadi, S. Escalera, and T. B. Moeslund, "Multi-scale hybrid vision transformer and sinkhorn tokenizer for sewer defect classification," *Automat. Construct.*, vol. 144, Dec. 2022, Art. no. 104614, doi: 10.1016/j.autcon.2022.104614.

[33] S. Sheynin, S. Benaim, A. Polyak, and L. Wolf, "Local-global shifting vision transformers," in *Proc. ICLR*, 2022, pp. 1–11.

[34] K.-W. Park and S.-B. Cho, "A vision transformer enhanced with patch encoding for malware classification," in *Intelligent Data Engineering and Automated Learning-IDEAL*. Manchester, U.K.: Springer, Nov. 2022, pp. 289–299.

[35] S. Seneviratne, R. Shariffdeen, S. Rasnayaka, and N. Kasthuriarachchi, "Self-supervised vision transformers for malware detection," *IEEE Access*, vol. 10, pp. 103121–103135, 2022.
[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[37] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: When to warp?" in *Proc. Int. Conf. Digit. Image Comput., Techn. Appl. (DICTA)*, Nov. 2016, pp. 1–6.
[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.
[39] A. Oliveira. (2019). *Malware Analysis Datasets: Top-1000 Pe Imports*. [Online]. Available: https://dx.doi.org/10.21227/004e-v304
[40] M. Alazab, S. Venkatraman, P. Watters, M. Alazab, and A. Alazab, "Cybercrime: The case of obfuscated malware," in *Proc. Int. Conf. e-Democracy Int. Conf. Global Secur., Saf., Sustainability*. Cham, Switzerland: Springer, 2012, pp. 204–211.
[41] N. Kumar and T. Meenpal, "Texture-based malware family classification," in *Proc. 10th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2019, pp. 1–6.
[42] M. M. Belal, T. Maitra, D. Giri, and A. K. Das, "Chaotic neural networks and farfalle construction based parallel keyed secure hash function," *Secur. Privacy*, vol. 5, no. 6, p. e259, Nov. 2022.
[43] Y. Tang, X. Qi, J. Jing, C. Liu, and W. Dong, "BHMDC: A byte and hex n-gram based malware detection and classification method," *Comput. Secur.*, vol. 128, May 2023, Art. no. 103118, doi: 10.1016/j.cose.2023.103118.
[44] H. Deng, C. Guo, G. Shen, Y. Cui, and Y. Ping, "MCTVD: A malware classification method based on three-channel visualization and deep learning," *Comput. Secur.*, vol. 126, Mar. 2023, Art. no. 103084, doi: 10.1016/j.cose.2022.103084.
[45] R. Chaganti, V. Ravi, and T. D. Pham, "Image-based malware representation approach with EfficientNet convolutional neural networks for effective malware classification," *J. Inf. Secur. Appl.*, vol. 69, Sep. 2022, Art. no. 103306, doi: 10.1016/j.jisa.2022.103306.
[46] J.-Y. Kim and S.-B. Cho, "Obfuscated malware detection using deep generative model based on global/local features," *Comput. Secur.*, vol. 112, Jan. 2022, Art. no. 102501, doi: 10.1016/j.cose.2021.102501.
[47] Z. Cui, L. Du, P. Wang, X. Cai, and W. Zhang, "Malicious code detection based on CNNs and multi-objective algorithm," *J. Parallel Distrib. Comput.*, vol. 129, pp. 50–58, Jul. 2019, doi: 10.1016/j.jpdc.2019.03.010.
[48] N. Bhodia, P. Prajapati, F. Di Troia, and M. Stamp, "Transfer learning for image-based malware classification," 2019, *arXiv:1903.11551*.
[49] Z. Ren, G. Chen, and W. Lu, "Malware visualization methods based on deep convolution neural networks," *Multimedia Tools Appl.*, vol. 79, nos. 15–16, pp. 10975–10993, Apr. 2020.
[50] Z. Cui, F. Xue, X. Cai, Y. Cao, G. Wang, and J. Chen, "Detection of malicious code variants based on deep learning," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3187–3196, Jul. 2018, doi: 10.1109/TII.2018.2822680.



MOHAMAD MULHAM BELAL received the bachelor's degree in electronic engineering with a specialization in computer engineering from Aleppo University, Syria, in 2017, and the M.Tech. degree in computer science and engineering from the School of Computer Engineering, KIIT (Deemed to be University), Bhubaneswar, India, in 2021. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, VIT-AP University, Amaravati, India. His research interests include information security, cloud computing security, and machine learning.

DIVYA MEENA SUNDARAM received the B.Tech. degree in information technology from the Vellore Institute of Technology (VIT), Vellore, India, in 2014, the M.E. degree in computer science and engineering from Anna University, Chennai, in 2016, and the Ph.D. degree from VIT University, in 2020. She was an Assistant Professor at the Jansons Institute of Technology, for a year. She was also an Assistant Professor at Jain University, Bengaluru, before moving to VIT-AP campus. With a total experience of 1.6 years, she has authored or coauthored 26 articles in peer-reviewed reputed journals and has more than ten international conference publications. Her reputed publications include research articles in peer-reviewed journals, namely, IEEE Access, *Neural Processing Letters*, *International Journal of Fuzzy Systems*, *Environmental Science and Pollution Research*, *Journal of Applied Remote Sensing*, and *Multimedia Tools and Applications*, and indexing at Thomson Reuters with an average impact factor of 8+. Her areas of interests include image processing and analysis, neural networks, fuzzy logic, machine learning, thermal imaging, and remote sensing. She received the Best Researcher Award for the past three years, from 2017 to 2020.