




Research Article

A Novel Malware Detection and Family Classification Scheme for IoT Based on DEAM and DenseNet

Changuang Wang ^{1,2}, Ziqiu Zhao,² Fangwei Wang ^{1,2} and Qingru Li ^{1,2}

¹Key Lab of Network & Information Security of Hebei Province, Shijiazhuang 050024, China

²College of Computer & Cyber Security, Hebei Normal University, Shijiazhuang 050024, China

Correspondence should be addressed to Fangwei Wang; fw_wang@hebtu.edu.cn and Qingru Li; liqingru2006@163.com

Received 20 November 2020; Revised 12 December 2020; Accepted 19 December 2020; Published 5 January 2021

Academic Editor: Athanasios V. Vasilakos

Copyright © 2021 Changuang Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the rapid increase in the amount and type of malware, traditional methods of malware detection and family classification for IoT applications through static and dynamic analysis have been greatly challenged. In this paper, a new simple and effective attention module of Convolutional Neural Networks (CNNs), named as Depthwise Efficient Attention Module (DEAM), is proposed and combined with a DenseNet to propose a new malware detection and family classification model. Based on the good effect of the DenseNet in the field of image classification and the visual similarity of the malware family on images, the gray-scale image transformed from malware is input into the model combined with the DEAM and DenseNet for malware detection, and then the family classification is carried out. The DEAM is a general lightweight attention module improved based on the Convolutional Block Attention Module (CBAM), which can strengthen the attention to the characteristics of malware and improve the model effect. We use the MalImg dataset, Microsoft malware classification challenge dataset (BIG 2015), and our dataset constructed by the two above-mentioned datasets to verify the effectiveness of the proposed model in family classification and malware detection. Experimental results show that the proposed model achieves 99.3% in terms of accuracy for malware detection on our dataset and achieves 98.5% and 97.3% in terms of accuracy for family classification on the MalImg dataset and BIG 2015 dataset, respectively. The model can reliably detect IoT malware and classify its families.

1. Introduction

Malware is a kind of software program designed to access a computer system and perform useless or harmful operations. It includes viruses, worms, Trojan horses, advertising software, spyware, blackmail software, and other types. These kinds of software will obtain confidential data, steal identity, hijack traffic and operating system, encrypt digital assets, and monitor users, which poses threats to users and operating systems. Malware is constantly challenging the network security situation with its continuously increasing growth rate and endless family types. According to the statistics of “malware threat situation report 2020” [1] released by Malwarebytes labs, in 2019, the detection of Windows malware on business endpoints increased by 13%. Malware detection and family classification technology is

still a development direction that cannot be ignored. Similarly, the Internet of Things (IoT) devices built on different processor architectures have increasingly become targets of adversarial attacks. Although there are many ways to detect malware on the Internet of Things [2, 3], we still need to make further efforts in this field.

Traditional malware detection and family classification use two kinds of malware analysis techniques: static analysis and dynamic analysis. Static analysis disassembles executable programs and analyzes and extracts the characteristic information of code without executing malware. In [4], sequential pattern mining technology is used to detect the maximum frequent pattern (MFP) of the opcode sequence for malware detection in the Internet of Things. In [5], the behavior sequence chain of some malware families is generated, and the similarity between the behavior sequence

chain and the sequence of the target process is calculated to detect and classify malware. In [6], malware is identified by combining normalized compression distance (NCD) with the compressibility rates of executables using decision forests. However, static analysis may consume a lot of time in useless code because the code analyzed is not necessarily the code of final execution. At the same time, the reliance of static analysis on disassembly technology also results in malware that can use various obfuscation techniques to hinder disassembly analysis. Some malware makes reverse engineering more complex by encrypting, packaging, and so on, which increases the difficulty of static analysis. Dynamic analysis is the extraction of feature information in the process of code execution, and the analyzed code is the actual execution code. In [7], malicious artifacts are extracted from memory through memory forensics technology, and malware detection is performed by combining the extracted malicious artifacts with the features extracted when executing malware files using dynamic analysis. In [8], the confused malware is detected by proper hook installation and real calculation of malware activity time in user and kernel. In [9], a graph repartitioning algorithm that uses the N-order subgraph (NSG) to convert API call graphs into fragment behaviors is proposed for malware detection and family classification. Besides, the “term frequency-inverse document frequency” (TF-IDF) and information gain (IG) were improved and used to extract the crucial N-order subgraph (CNSG). However, dynamic analysis of one execution process can only obtain a single path behavior, and some malware has multiple execution paths. At the same time, dynamic analysis has certain risks due to the actual execution of the program. With the development of neural networks in recent years, static analysis and dynamic analysis are often combined with neural networks for malware detection and family classification. In [10], the bigram model is used to represent the opcode, the frequency vector is used to represent the API call, and then convolutional neural network and backpropagation neural network are used to embed features based on opcodes and APIs for malware detection and family classification. In [11], based on the behavior of malware, a classification method based on malware type was proposed, and LSTM was used for a new dataset developed for Windows operating system based on API calls.

The main purpose of this paper is to combine the attention module and convolutional neural network to better perform malware detection and family classification. In our framework, we convert the malware samples into gray-scale images and then apply DenseNet with Depthwise Efficient Attention Module to the images. In this process, DEAM can generate feature attention maps to strengthen the attention to malware features, to improve the effectiveness of detection and family classification. The main contributions of our work are as follows.

This paper proposes a new general lightweight attention module, DEAM, which can be widely used to improve the performance of CNNs, while not increasing the amount of calculation. It consists of both the Improved Efficient Channel Attention (IECA) and a new spatial attention

mechanism, Depthwise Spatial Attention (DSA). We replace the SENet in the original channel attention mechanism structure in CBAM with ECA-Net to get the IECA of the proposed model. The DSA is constructed by using Depthwise Convolution. We combine the DSA module and the DenseNet for malware detection and family classification.

The proposed model performs well on the MalImg dataset, the BIG 2015 dataset, and the dataset built by us and can effectively perform malware detection and family classification.

The rest of the paper is organized as follows: Section 2 discusses the related work concerning different techniques such as visualization, the structure of the CNNs, attention mechanism in malware detection, and classification. Section 3 presents our proposed model in detail. The performance of our algorithm is evaluated in Section 4. Section 5 summarizes our work and put forward some suggestions for future work.

2. Related Work

2.1. Malware Visualization. In this paper, we use the method proposed in the literature [12] to convert malware into gray-scale images. We convert each byte (8-bit binary or 2-bit hexadecimal) in the PE file into a pixel, and the value of each pixel is in the range of [0, 255] (0: black, 255: white). The height of the image is determined by the size of the PE file, as shown in Table 1.

Through malware images, we can find that the images of malware from the same family are visually similar, but there are large visual differences between different malware families. Besides, the difference also exists between benign software and malware, as shown in Figure 1. Converting malware into images can help us perform malware analysis. After being converted into images, different parts of the file can be easily distinguished so that we can find the functional parts of the malware.

Converting malware into images for detection and family classification has become a common practice in recent years. In [13], the memory data dump file was converted into a gray-scale image, and the histogram of gradient (HOG) extraction function was used to effectively classify the malware. In [14], a new hybrid model based on image analysis was proposed, which uses similarity mining and deep learning architecture to accurately identify and classify confusing malware. Inspired by the visual similarity between malware samples of the same family, a file-agnostic deep learning method is proposed for malware classification [15]. Through a set of discriminative patterns extracted from the visualized image of the malware, the malware is effectively divided into multiple families. In [16], based on the visual similarity between malware in the same family, a suggestion of directly performing binary texture analysis on gray-scale images of malware executable files was proposed. This technology derives a new combination of second-order statistical texture features based on the first-order and gray-level cooccurrence matrix (GLCM) on the visualized malware to perform confusion and unbalanced malware classification.

TABLE 1: Image height for several file sizes.

File size	Image height
<10 kB	32
10 kB–30 kB	64
30 kB–60 kB	128
60 kB–100 kB	256
100 kB–200 kB	384
200 kB–500 kB	512
500 kB–1000 kB	768
>1000 kB	1024

2.2. Structure of the CNNs. The Convolutional Neural Networks (CNNs) have greatly promoted the development of image classification with their excellent performance. Recently, in order to improve the performance of Convolutional Neural Networks, researchers have made many changes in three aspects: depth, width, and cardinality. Starting from LeNet [17], the pioneering work of the CNNs, and then the outbreak after AlexNet [18], the structure of CNNs has been becoming deeper and deeper to achieve richer representations. VGGNet [19] proved that increasing the depth of the network can affect the final performance of the network to a certain extent. ResNet [20] introduced a shortcut to make the network have a certain identity mapping ability, and strengthen the correlation of gradients between the layers of the network. GoogLeNet [21] proved that width is another important factor to improve model performance. DenseNet [22] further deepened the idea of ResNet, applied a shortcut to the entire network, realized the dense connection of the network, and strengthened the connection between features of each layer. Image classification technology based on DenseNet has recently been applied to various fields [23–25]. However, as the model continues to be expanded in depth, width, and base, the amount of its calculation is also increasing. In order to achieve a better balance between the performance and cost of the model, it is more possible to build a universal bionic mechanism in the deep learning model than to pile up more nonlinear layers.

2.3. Attention Mechanism. The attention mechanism is a deep learning technology that originated from the study of human vision and has been widely used in natural language processing [26, 27], recommendation systems, and image classification [28, 29]. It mimics the characteristics of the human visual system that selectively focuses on the salient parts, and improves the efficiency of the model by dynamically selecting important features. It can be found from the development in recent years that the attention mechanism has become a common method to enhance the effect of CNNs. The attention map obtained by the attention mechanism from CNNs shows specific areas, which represent the features being focused on.

SENet [30] first proposed an effective channel attention learning block and achieved good performance, proving that attention can improve the expressiveness of the network by enhancing important features and suppressing unnecessary features. In [31], the malware is converted into a gray-scale

image and then input into the model combined with SENet [30] and CNN for malware analysis and family classification. After that, the attention module is developed from two aspects: the enhancement of feature aggregation or the combination of channel and spatial attention. GSoP [32] introduced a second-order pool to achieve more effective feature aggregation. CBAM [33] proposed a general attention module for CNNs, which uses max pooling and average pooling to aggregate features and uses the aggregated features for sequential channel attention mechanism (using SENet [30]) and spatial attention mechanism. ECA-Net [34] improved SENet [30] according to the idea of no dimensionality reduction and lightweight and improved the effect while reducing the parameters. ADCM [35] integrates dropout into the attention mechanism according to the idea of lightweight and improves CBAM [33]. In addition, many works use improved attention mechanisms to improve the effect of CNNs [36, 37].

Based on the CBAM [33] framework, this paper improves the channel attention mechanism inside and creates a new spatial attention mechanism. A new general lightweight attention module called Depthwise Efficient Attention Module is proposed.

3. Proposed Model

In order to better perform malware detection and family classification of malware, we proposed a new method based on DenseNet and the attention mechanism. In this section, we will introduce the proposed model in detail. The proposed model is composed of DenseNet and DEAM. Based on DenseNet-121, we construct a DenseNet suitable for the proposed model. DEAM is composed of Improved Efficient Channel Attention (IECA) and Depthwise Spatial Attention (DSA). First, we introduce the architecture of DenseNet. Then the proposed IECA and the DSA are described, respectively. Finally, the entire flowchart of our model for malware detection and family classification is represented.

3.1. Structure of the DenseNet. The DenseNet model is a deep learning model developed on the ResNet. In recent years, DenseNet has achieved better results in the field of image classification. The basic idea of ResNet and DenseNet is the same; however, DenseNet establishes a dense connection between all the previous layers and the latter, and it realizes feature reuse through the connection of features on the channel. These features make DenseNet achieve better performance than ResNet with fewer parameters and computational costs and alleviate gradient vanishing problems.

The DenseNet is mainly composed of DenseBlock and Transition layer. DenseBlock adopts a radical dense connection mechanism; that is, all layers are connected to each other. Specifically, each layer accepts the output from all the previous layers as its additional input, as shown in Figure 2.

In DenseBlock, each layer has the same size and each layer is concatenated with all previous layers in the channel dimension. For a network with L layer, DenseBlock contains

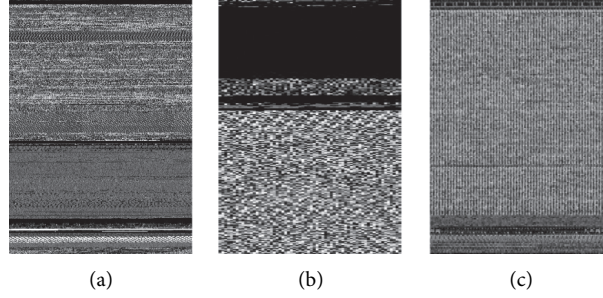


FIGURE 1: Malware samples from different families and benign samples are visualized as gray-scale images. Note the texture differences of different samples: (a) malware sample (Adialer.C); (b) malware sample (Agent.FYI); (c) benign sample.

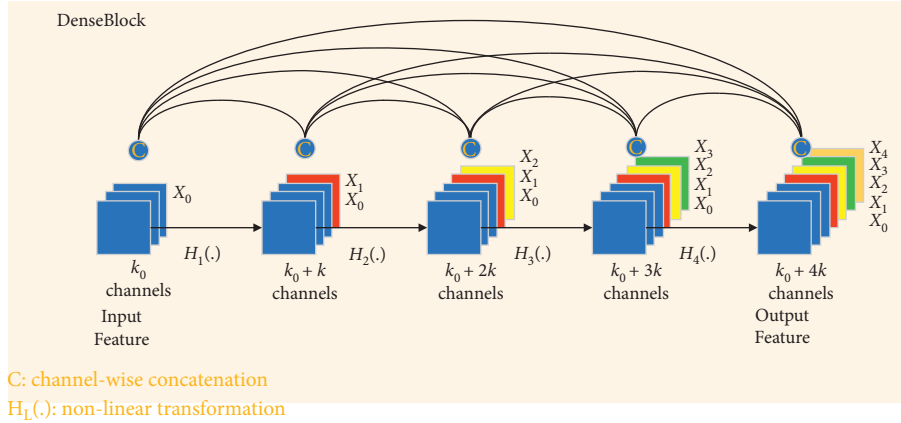


FIGURE 2: Implementation mechanism of DenseBlock.

a total of $L(L+1)/2$ connections. The input of the layer L is as follows:

$$x_L = H_L([x_1, x_2, \dots, x_{L-2}, x_{L-1}]), \quad (1)$$

where L represents the number of layers. $H_L(\dots)$ represents nonlinear transformation, which is a combination of Batch Normalization (BN), ReLU, Pooling, and Conv operations. In this paper, the common DenseNet-B structure is utilized, and the bottleneck layer is used to reduce the amount of calculation; that is, the structure BN + ReLU + 1×1 Conv + BN + ReLU + 3×3 Conv is adopted in this paper. Each layer in DenseBlock outputs k feature maps after convolution, that is, the number of convolution kernels. If we set the channel number of input DenseBlock as k_0 , then the input channel number of L layer is $k_0 + k(L-1)$. Here, the final convolution of each layer is k , and k is called the growth rate. In DenseBlock, with the increase in the number of layers, the number of input channels will be larger and larger.

Since the input size of the model after passing through a DenseBlock remains unvaried, the channel dimension will continue to increase. Therefore, dimension reduction is necessary to reduce computational complexity. The Transition layer is mainly composed of a 1×1 convolution and 2×2 AvgPooling or MaxPooling, and its structure is BN + ReLU + 1×1 Conv + 2×2 AvgPooling. It connects two

adjacent DenseBlocks and reduces the dimensionality of the output of the DenseBlock.

Now the commonly used DenseNet frameworks are DenseNet-121, DenseNet-169, DenseNet-201, and DenseNet-264. The DenseNet in our proposed model is based on DenseNet-121. Table 2 shows a comparison between DenseNet-121 and the DenseNet in our proposed model.

3.2. Depthwise Efficient Attention Module. The DEAM we proposed follows the framework of CMBA [33] and consists of two parts, IECA and DSA. For an input feature map $M \in R^{C \times H \times W}$ (where C denotes channel, H denotes height, and W denotes width), DEAM calculates the relationship between the channels of the feature map through IECA to obtain a 1-dimensional channel attention map $M_C \in R^{C \times 1 \times 1}$ to focus on important features on the image. Then, DEAM calculates the 3-dimensional spatial attention map $M_S \in R^{C \times H \times W}$ of the feature map through DSA and pays attention to the position of the feature on the image. The calculation process in the DEAM is as follows:

$$\begin{aligned} M' &= M_C(M) \otimes M, \\ M'' &= M_S(M') \otimes M', \end{aligned} \quad (2)$$

where \otimes denotes elementwise multiplication.

According to [33], in the DEAM, we connect the two attention mechanisms serially and put IECA in the front of

TABLE 2: The structure of DenseNet in the proposed model and its comparison with DenseNet-121.

Layers	Our DenseNet ($k=12$)	DenseNet-121 ($k=32$)
Convolution	3 * 3 conv, stride 1	7 * 7 conv, stride 2
Pooling	2 * 2 max pool, stride 2	3 * 3 max pool, stride 2
DenseBlock(1)	$\begin{pmatrix} 1 * 1 \text{ conv} \\ 3 * 3 \text{ conv} \\ 1 * 1 \text{ conv} \end{pmatrix} * 3$	$\begin{pmatrix} 1 * 1 \text{ conv} \\ 3 * 3 \text{ conv} \end{pmatrix} * 6$
Transition	2 * 2 ave pool, stride 2	
DenseBlock(2)	$\begin{pmatrix} 1 * 1 \text{ conv} \\ 3 * 3 \text{ conv} \\ 1 * 1 \text{ conv} \end{pmatrix} * 6$	$\begin{pmatrix} 1 * 1 \text{ conv} \\ 3 * 3 \text{ conv} \end{pmatrix} * 12$
Transition	2 * 2 ave pool, stride 2	
DenseBlock(3)	$\begin{pmatrix} 1 * 1 \text{ conv} \\ 3 * 3 \text{ conv} \\ 1 * 1 \text{ conv} \end{pmatrix} * 12$	$\begin{pmatrix} 1 * 1 \text{ conv} \\ 3 * 3 \text{ conv} \end{pmatrix} * 24$
Transition	2 * 2 ave pool, stride 2	
DenseBlock(4)	$\begin{pmatrix} 1 * 1 \text{ conv} \\ 3 * 3 \text{ conv} \end{pmatrix} * 8$	$\begin{pmatrix} 1 * 1 \text{ conv} \\ 3 * 3 \text{ conv} \end{pmatrix} * 16$
Pooling	GlobalAveragePooling	
	Softmax	

DEAM to get the best effect. Through experimental comparisons, when DEAM is put behind the last DenseBlock of DenseNet, the system can achieve the best effect. Since DenseNet is the connection of all layers, the input of each layer is the superposition of all the previous layers. Adding a DEAM behind each DenseBlock will recalculate the calculated value and create a lot of useless overhead. In addition, each DEAM focuses on different features. If the DEAM is in front of or behind each DenseBlock, they may interfere with each other to reduce the effect of the model. Figure 3 describes the process of each attention map, and the detailed information of each attention mechanism is described below.

3.2.1. Improved Efficient Channel Attention. In the mechanism of channel attention, we consider which features we should pay attention to. Each channel of the feature map is regarded as a feature detector [38]; however, not every channel is very useful for image recognition. By calculating the probability of different channels, the channel attention will be focused on the main features of the image. Therefore, through the channel attention mechanism, we can better extract the representative features of malware images and improve the efficiency of malware detection and family classification.

The attention mechanism has been widely used to improve the performance of CNNs, among which the more representative ones are SENet [30] and CBAM [33]. However, most of the attention mechanisms are dedicated to complicating themselves to achieve better performance. ECA-Net [34] improved SENet model by lightweight without dimensionality reduction, and the important role of no dimensionality reduction for the attention mechanism is proved. ECA-Net proposes a 1-dimensional convolution to achieve a local cross-channel interaction strategy, which

reduces model complexity while improving performance. The formula is as follows:

$$\omega = \sigma(C1D_k(y)), \quad (3)$$

where $C1D$ denotes 1-dimensional convolution, k denotes convolution kernel size of 1-dimensional convolution $y \in R^{C \times 1 \times 1}$, ω denotes channel attention map, and σ denotes Sigmoid function. Meanwhile, a method of adaptively selecting the size of a 1-dimensional convolution kernel is proposed to determine the coverage rate of local cross-channel interaction. The formula is as follows:

$$k = \psi(C) = \left\lfloor \frac{\log_2(C)}{\gamma} + \frac{b}{\gamma|_{\text{odd}}} \right\rfloor, \quad (4)$$

where $|t|_{\text{odd}}$ denotes the odd number closest to t , γ and b are set to 2 and 1, respectively, and C denotes the channel number of the input feature map. This improvement significantly reduces the parameters of the channel attention mechanism and enhances the computational efficiency of the model.

In this paper, we use ECA-Net to replace the SENet in the channel attention mechanism structure in CBAM in order to achieve the effect without local dimensionality reduction, and then we get the IECA. We use the max pool and average pool to compress the input feature map in the channel space in order to effectively calculate the channel attention. The average pool gathers spatial information, and the max pool gathers distinctive object features. Two spatial context descriptors (M_{avg}^C and M_{max}^C) which represent average pool feature and max pool feature are output from average pool and max pool, respectively. The two spatial context descriptors are combined into a feature vector $F_0 \in R^{C \times 1 \times 1}$ by element summation, and the combined feature vector is transferred into a 1-dimensional convolution. The size k of the 1-dimensional convolution

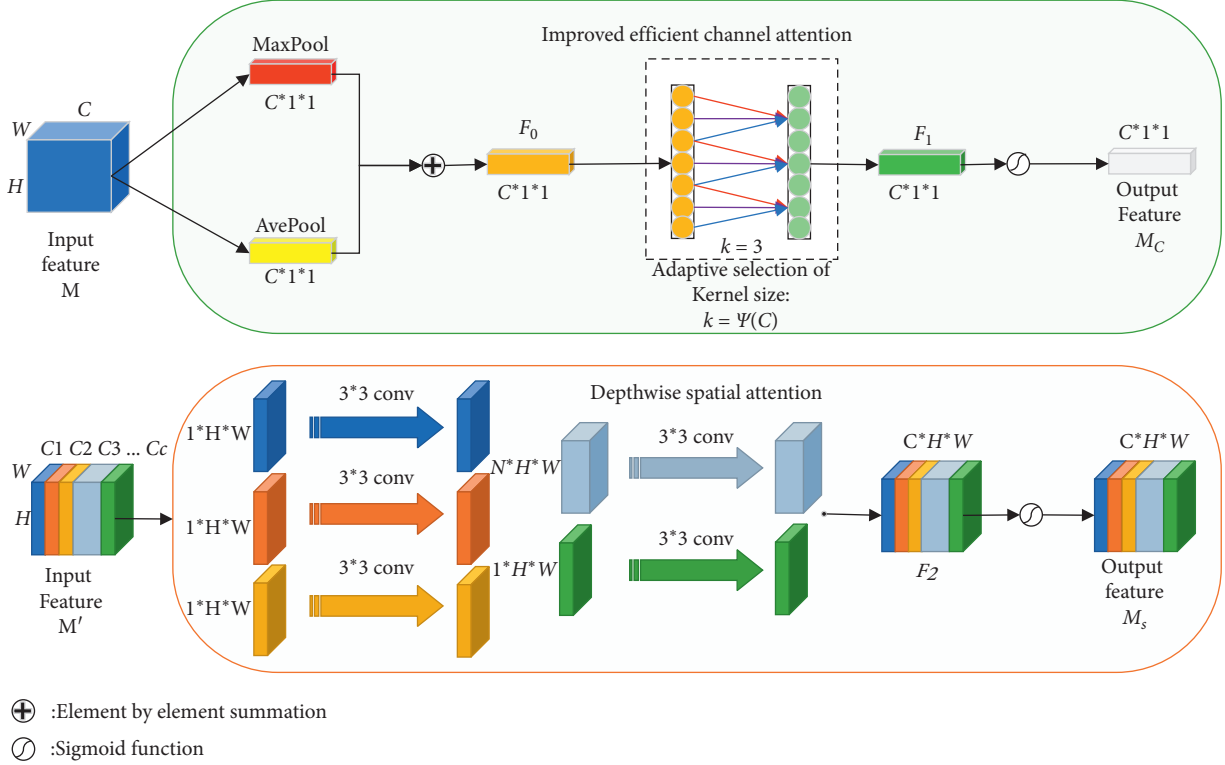


FIGURE 3: Diagram of each attention submodule of the EDAM. As illustrated, the IECA submodule combines the maximum pool output and average pool output into a 1-dimensional convolution to generate a 1-dimensional channel attention map; the DSA submodule uses the Depthwise Convolution to generate the 3-dimensional space attention map.

kernel is obtained by the adaptive selection, and the Sigmoid function is used for the eigenvector of a 1-dimensional convolution output $F_1 \in R^{C \times 1 \times 1}$ to obtain a 1-dimensional channel attention map $M_C \in R^{C \times 1 \times 1}$. The formula is as follows:

$$\begin{aligned} M_C(M) &= \sigma(\text{conv1} D(\text{Avg Pool}(M) + \text{Max Pool}(M))), \\ M_C(M) &= \sigma(C1D_k(M_{\text{avg}}^C + M_{\text{max}}^C)), \end{aligned} \quad (5)$$

where + denotes element summation.

3.2.2. Depthwise Spatial Attention. Different from the channel attention mechanism, the spatial attention mechanism pays attention to which position on the feature detector is meaningful and which part is a supplement to the channel attention mechanism. The spatial attention mechanism will calculate the probability of different positions on the feature map and focus on the meaningful positions on the feature map. Therefore, through the spatial attention mechanism, we can better extract the representative features of malware images and improve the efficiency of malware detection and family classification.

The spatial attention mechanism of CMBA [33] first compresses the feature map with the max pool and the average pool along the channel axis [39] and then connects their outputs to generate an effective feature descriptor. Convolution is used for the feature descriptor to generate a 2-dimensional spatial attention map. DSA in this paper is a

new spatial attention mechanism that is constructed based on the idea of no dimensionality reduction in ECA-Net [34]. In ECA-Net, it has been proved that avoiding dimensionality reduction is very important for the attention mechanism. DSA uses Depthwise Convolution to calculate the 3-dimensional spatial attention map of the feature map without dimensionality reduction. Depthwise Convolution is a special form of Group Convolution when the number of groups is equal to the number of channels. Depthwise Convolution divides the input features into different groups according to the number of channels and convolves each group separately. In IECA, we only replace the part of SENet [30] to achieve the effect of no local dimensionality reduction in the channel attention mechanism. In DSA, we construct a new spatial attention mechanism through Depthwise Convolution, abandoning the dimensionality reduction of the max pool and average pool along the channel axis in the CMBA, and achieve the effect of no dimensionality reduction. Depthwise Convolution can obtain a prominent information area from each channel, which is more comprehensive than applying the convergence operation along the channel axis [39]. We will describe the detailed operation below.

We apply Depthwise Convolution on the input feature map and use the Sigmoid function on the output feature descriptor $F_2 \in R^{C \times H \times W}$ to get a 3-dimensional spatial attention map $M_S \in R^{C \times H \times W}$. The formula is as follows:

$$M_S(M') = \sigma(\text{Depthwise Conv2} D(M')), \quad (6)$$

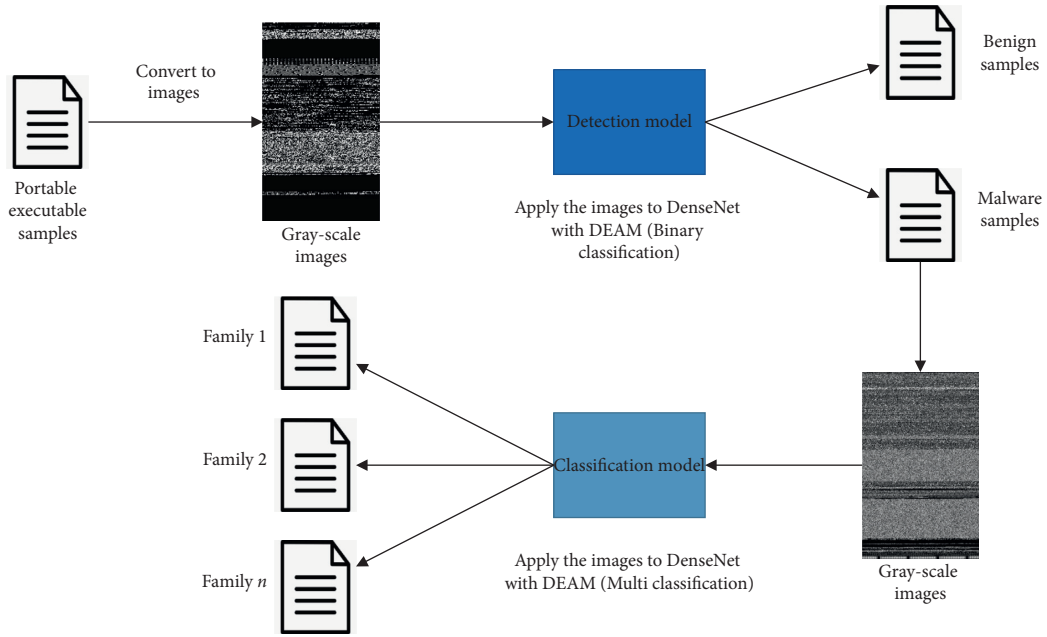


FIGURE 4: Block diagram of malware detection and classification.

TABLE 3: MalImg: distribution of samples.

No.	Family	Number of samples
1	Adialer.C	122
2	Agent.FYI	116
3	Allaple.A	2949
4	Allaple.L	1591
5	Alureon.gen!J	198
6	Autorun.K	106
7	C2LOP.gen!g	200
8	C2LOP.P	146
9	Dialplatform.B	177
10	Dontovo.A	162
11	Fakerean	381
12	Instantaccess	431
13	Lolyda.AA1	213
14	Lolyda.AA2	184
15	Lolyda.AA3	123
16	Lolyda.AT	159
17	Malex.gen!J	136
18	Obfuscator.AD	142
19	Rbot!gen	158
20	Skintrim.N	80
21	Swizzor.gen!E	128
22	Swizzor.gen!I	132
23	VB.AT	408
24	Wintrim.BX	97
25	Yuner.A	800
		9339

where Depthwise Conv2D denotes Depthwise Convolution.

3.3. The Process of Detecting and Classifying Malware. The whole process of detecting and classifying malware is described as follows. First, the PE file is converted into a gray-scale image using the method in the literature [12]. Second,

TABLE 4: BIG 2015: distribution of samples.

No.	Family	Number of samples
1	Ramnit	1541
2	Lollipop	2478
3	Kelihos_ver3	2942
4	Vundo	475
5	Simda	42
6	Tracur	751
7	Kelihos_ver1	398
8	Obfuscator.ACY	1228
9	Gatak	1013
		10868

TABLE 5: Precision, recall, and F1 score of our model on the constructed dataset.

	Malware	Benign	Precision	Recall	F1 score
Malware	204	0	0.986	1	0.992
Benign	3	214	1	0.986	0.993
Macro			0.993	0.993	0.993

TABLE 6: Precision, recall, and F1 score of DenseNet on the constructed dataset.

	Malware	Benign	Precision	Recall	F1 score
Malware	201	3	0.995	0.985	0.990
Benign	1	216	0.986	0.995	0.990
Macro			0.991	0.990	0.990

the converted gray-scale image is applied to a malware detection model, which consists of DEAM and DenseNet. The model is trained using the gray-scale images of known benign software samples and malware samples, as well as

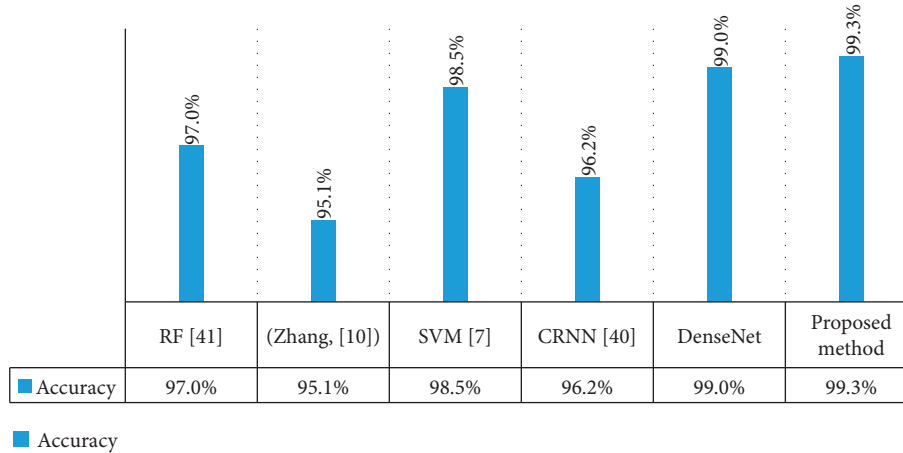


FIGURE 5: Accuracy comparison of our model with recent works in detection.

their corresponding labels. The trained detection model can effectively distinguish the malware from benign software. Then, the gray-scale image of malware is applied to the malware family classification model which consists of DEAM and DenseNet. The model is trained using gray-scale images of known malware samples and their labels representing the family of each malware sample. The trained family classification model can effectively identify malware families. The whole process is shown in Figure 4.

4. Experiments

4.1. Datasets and Evaluation Criterion. The datasets used for the evaluation of the classification results of malware families in this article are the Mallmg dataset from [12] and the BIG 2015 dataset provided by Microsoft for the Big Data Innovators Gathering Anti-Malware Prediction Challenge. The Mallmg dataset is a large-scale unbalanced Windows malware gray-scale image dataset which contains 25 malware families and a total of 9339 malware gray-scale image samples, as shown in Table 3. The malware families in Mallmg dataset include worm, Trojan horse, backdoor, and rogue software.

We only use the labeled training set of the BIG 2015 dataset, which contains 9 malware families and a total of 10868 malware samples, as shown in Table 4. Each sample of the dataset has a hexadecimal representation of its binary content and its corresponding assembly file. Both the Mallmg dataset and the BIG 2015 dataset are benchmark datasets used in many recent works.

Due to the lack of public datasets for detection, we use our own constructed dataset for indirect comparison with the work of others. We merged the Mallmg dataset and the BIG 2015 dataset and randomly selected the same number of malware samples from each of the 34 malware families in the merger. Then, we constructed a 1:1 detection dataset with the extracted 1087 malware samples and the collected 1087 benign software samples. The diversity of malware families in the dataset ensures the generalization ability of the detection model, so as to avoid the reduction of model

generalization performance caused by overfitting when using unbalanced data to train the model.

In order to test the generalization performance of our model, we divided the dataset into the training set, validation set, and test set at a ratio of 6:2:2 and repeated each experiment 5 times to reduce experimental errors. The training set is used to train the model, the validation set is used to adjust the performance of the model, and the test set is put into the trained model to test the performance of the model. Our experiment is based on the TensorFlow 2.0 framework. The Adam optimizer and categorical_crossentropy are used in our experiments. Besides, such parameters as accuracy, recall, and F1 score are also used as performance indicators to select the best model in detection and family classification. This is because when there is an imbalance between different classes, the accuracy rate can only reflect the overall prediction level. It ignores the prediction ability of a small number of classes. Sometimes, it can still get a higher level of classification accuracy when there are errors in a small number of classes or key classes. The precision is relative to the prediction results and indicates the correct number of the samples whose predictions are positive. The recall rate is relative to the sample, that is, how many positive samples are correctly predicted. The F1 score combines the precision and recall results.

The precision is calculated as follows:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad (7)$$

where TP is the true positive number and FP is the false positive number.

The recall is obtained as follows:

$$\text{Recall} = \frac{TP}{TP + FN}, \quad (8)$$

where FN is the false negative number.

The F1 score is a weighted harmonic mean of precision and recall, as follows:

$$F1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (9)$$

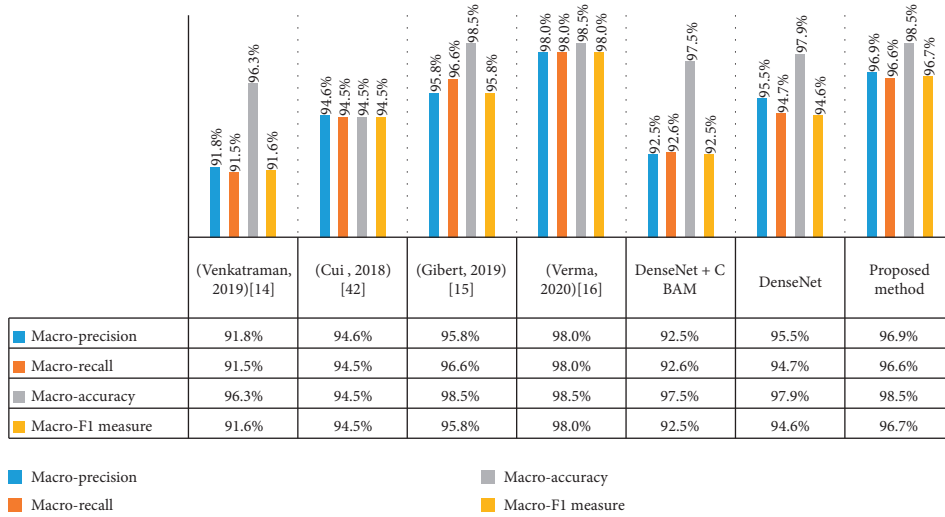


FIGURE 6: Comparison of our model with recent works on MalImg dataset.

We derive the values of the binary classification task (detection) normally. For multiclassification tasks (family classification), we obtain the precision, recall, and F1 scores of each family separately. After that, the macro-precision, macro-recall, and macro-F1 are calculated by averaging the sum of the evaluation indicators of each family (the macro-average gives each family the same weight), respectively.

4.2. Malware Detection. We conduct malware detection experiments on the constructed dataset. Tables 5 and 6 show the obtained detection results in the form of a 2×2 confusion matrix, as well as the precision, recall, and F1 score values of each family. For the proposed model, the results of the accuracy, precision, recall, and F1 score are all 99.3%. The DenseNet without DEAM has an accuracy of 99.0%, a precision of 99.1%, a recall of 99.0%, and an F1 score of 99.0% on our constructed dataset. It can be found that DEAM almost does not improve the performance of CNNs in terms of detection, but the numbers of wrong predicted samples in the two experiments are only 3 and 4, respectively. Therefore, we believe that DenseNet itself already has high detection capabilities, and adding DEAM can no longer improve the performance of CNNs. Figure 5 gives an indirect comparison between our model and recent works, [7, 10, 40, 41] which proves that the proposed model is superior in detection compared with existing methods.

4.3. Malware Family Classification

4.3.1. MalImg Dataset. Our model is trained by reducing the image size in the MalImg dataset to $192 * 192$ pixels. The smaller image size cannot retain all important information (that is, loss of discriminative information about a family), while a higher value will only increase the calculation time while not improving the overall accuracy. Tables 7 and 8 give the classification results in the form of a 25×25 confusion matrix, as well as the precision, recall, and F1 score values of each family. The proposed model achieves very good results:

the accuracy is 98.5%, the precision is 96.9%, the recall is 96.6%, and the F1 score is 96.7%. These performance indicators show that the method achieves better classification and lower misclassification. On the MalImg dataset, DenseNet without DEAM has an accuracy of 97.9%, a precision of 95.5%, a recall of 94.7%, and an F1 score of 94.6%. Figure 6 shows the comparison between our model and recent work on the MalImg dataset. Experimental results show that our model has the same accuracy as that of in the literature [16], but other performance indicators are slightly lower than those in the literature [16]. Compared with other recent works, [14, 15, 42] our model has improved performance in malware family classification and robustness in classification imbalance.

The MalImg dataset contains many samples processed through obfuscation techniques, such as packaging and encryption. Among them, Yuner.A, VB.AT, Malex.gen!J, Autorun.K, and Rbot!gen families use the same packaging technology, UPX, which makes them have similar structures, and it is difficult to distinguish them. However, our model classifies the Yuner.A with 100% accuracy; the F1 scores of Malex.gen!J and Rbot!gen are 97.4% and 99.9%, and the F1 scores of VB.AT and Autorun.K are also 93.6% and 95.7%. Allapple encrypts the code part in several layers using a random key. Our model classifies Allapple.A and Allapple.L at a rate of 100%. This proves that our model is robust in both packaging and encryption. Meanwhile, Swizzor.gen!E and Swizzor.gen!I that belong to the same family variants are also classified with the accuracy of 100%.

It can be seen from the comparison of Tables 7 and 8 that our model slightly reduces the classification effect in each category compared with DenseNet without DEAM. For example, the F1 score of Fakerean has dropped from 1 to 98.4%. However, it greatly improves the classification accuracy of Wintrim.BX, Lolyda.AA2, and Lolyda.AA3. The F1 score of Wintrim.BX is raised from 75% to 88.1%, the F1 score of Lolyda.AA2 is raised from 45.4% to 81.4%, and the F1 score of Lolyda.AA3 is raised from 66.6% to 76%, thus improving the overall classification accuracy. This proves

TABLE 9: Precision, recall, and F1 score of our model on BIG 2015 dataset.

	Ramnit	Lollipop	Kelihos_ver3	Vundo	Simda	Tracur	Kelihos_ver1	Obfuscator.ACY	Gatak	Precision	Recall	F1 score
Ramnit	295	2	0	0	0	4	1	6	1	0.973	0.954	0.964
Lollipop	0	491	0	0	0	0	0	2	3	0.985	0.989	0.987
Kelihos_ver3	0	0	588	0	0	0	0	0	0	0.998	1.000	0.999
Vundo	0	1	0	91	0	1	1	1	0	0.957	0.957	0.957
Simda	0	0	0	1	7	0	0	0	0	0.875	0.875	0.875
Tracur	2	2	0	1	0	140	1	0	4	0.945	0.933	0.939
Kelihos_ver1	0	0	0	0	0	0	76	0	3	0.950	0.962	0.955
Obfuscator.ACY	6	1	1	2	1	3	1	227	3	0.957	0.926	0.941
Gatak	0	1	0	0	0	0	0	1	200	0.934	0.990	0.961
Macro										0.953	0.954	0.954

TABLE 10: Precision, recall, and F1 score of DenseNet on BIG 2015 dataset.

	Ramnit	Lollipop	Kelihos_ver3	Vundo	Simda	Tracur	Kelihos_ver1	Obfuscator.ACY	Gatak	Precision	Recall	F1 score
Ramnit	285	8	0	0	1	1	3	11	0	0.976	0.922	0.948
Lollipop	0	495	0	1	0	0	0	0	0	0.948	0.997	0.972
Kelihos_ver3	0	0	588	0	0	0	0	0	0	0.989	1.000	0.994
Vundo	0	2	0	90	0	1	0	1	1	0.957	0.947	0.952
Simda	0	0	0	1	5	0	1	0	1	0.833	0.625	0.714
Tracur	3	2	2	1	0	135	2	0	5	0.978	0.900	0.937
Kelihos_ver1	0	1	2	0	0	0	76	0	0	0.915	0.962	0.938
Obfuscator.ACY	4	8	2	0	0	1	1	224	5	0.941	0.914	0.927
Gatak	0	6	0	1	0	0	0	2	193	0.941	0.955	0.948
Macro										0.942	0.914	0.926

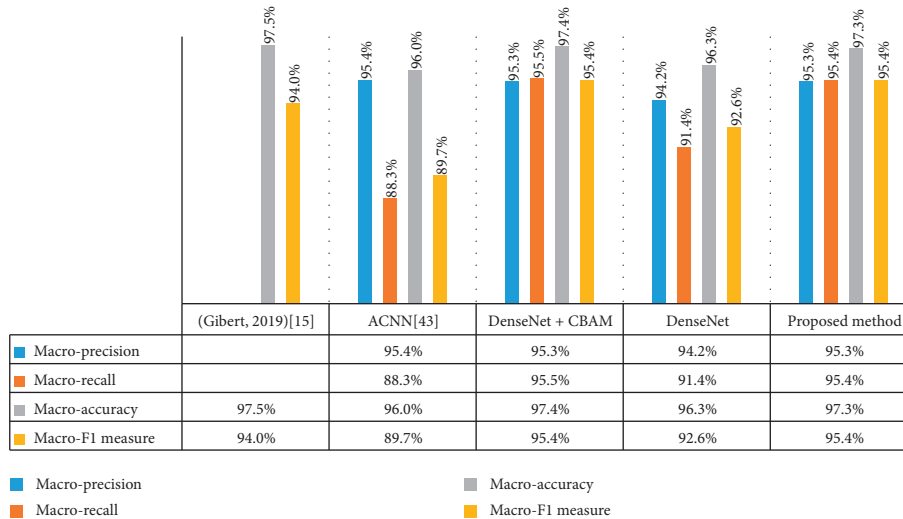


FIGURE 7: Comparison of our model with recent works on BIG 2015 dataset.

that our proposed DEAM has an improving effect on CNNs. Besides, in the entire model, the original CBAM block has 4,935 parameters, while our DEAM has only 1,935 parameters, almost reduced to one-third of CBAM parameters. For the 346,293 parameters in the entire DenseNet model, there is almost no increase in computational consumption.

On the MallImg dataset, the model using CBAM took Swizzor.gen!I for Obfuscator. AD with a rate of 100% in 5 experiments causing a significant drop in the classification effect, which reduced the performance of DenseNet.

4.3.2. BIG 2015 Dataset. The processing method of the BIG 2015 dataset is similar to that of the MallImg dataset. We downsample the gray-scale images. Tables 9 and 10 give the obtained classification results in the form of a 9×9 confusion matrix, as well as the precision, recall, and F1 measurement values of each family. The accuracy of our model on the BIG 2015 dataset is 97.3%, the precision is 95.3%, the recall is 95.4%, and the F1 score is 95.4%. DenseNet without DEAM has an accuracy of 96.3%, a precision of 94.2%, a recall of 91.4%, and an F1 score of 92.6% on the BIG 2015 dataset. Figure 7 shows a comparison of our model with recent works [15, 43] on the BIG 2015 dataset. It can be seen from Figure 7 that our model has improved precision, recall rate, and F1 score compared with [15], which proves that our model can better classify malware families. Our DEAM and CBAM have basically the same classification effect on the BIG 2015 dataset. However, the parameters used by DEAM are much less than those of CBAM, which can effectively improve the calculation efficiency. Based on the experimental results on the MallImg dataset and the BIG 2015 dataset, we can prove that the proposed DEAM has a better effect than CBAM. Through the comparison between Tables 9 and 10, we can see that the addition of DEAM has increased the classification effect of the model on multiple classes, especially the F1 score on Simda increasing from 71.4% to 87.5%. Therefore, it is further verified that DEAM improves the

performance of CNNs. There is a certain gap between the effect on BIG 2015 dataset and that on MallImg dataset. We think that this is due to the larger texture gap between the same family samples in BIG 2015. In this paper, we only use the global image and do not further process the gray-scale image of malware.

5. Conclusion

This paper proposes a new lightweight and effective convolutional neural network attention module that is defined as DEAM, and combines it with the DenseNet for malware detection and family classification. The proposed method, which is first used in the detection model, converts executable files into gray-scale images, and then the detected malware is used in the family classification model to distinguish different malware families. Experimental results show that the number of DEAM parameters is only one-third of the CBAM parameters, so the DEAM can reduce the attention module parameters and improve the computational efficiency of the model. Besides, it is better than CBAM in performance, which helps to improve the performance of CNNs. The presented model performs well in malware detection and family classification, and it also shows robustness to code confusion and class imbalance problems.

Although the proposed method has good performance in malware detection and family classification, it still needs improvements. For example, our method directly uses the original gray-scale image of the malware in the model and does not process the gray-scale image of the malware. In the future, we will explore these issues to further improve performance.

Data Availability

The MallImg dataset can be obtained from http://vision.ece.ucsb.edu/~lakshman/malware_images/album/. The BIG

2015 dataset can be obtained from <https://www.kaggle.com/malware-classification/data>.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grants No. 61572170 and No. 61672206, Natural Science Foundation of Hebei Province of China under Grant No. F2019205163, Science Foundation of Returned Overseas of Hebei Province of China under Grant No. C2020342, Science Foundation of Department of Human Resources and Social Security of Hebei Province under Grant No. 201901028, and Natural Science Foundation of Hebei Normal University under Grant No. L072018Z10.

References

- [1] 2020 State Of Malware Report, 2020, https://resources.malwarebytes.com/files/2020/02/2020_State-of-Malware-Report.pdf.
- [2] D. Vasan, M. Alazab, S. Venkatraman, J. Akram et al., “Cross-architecture IoT malware detection based on neural network advanced ensemble learning,” *IEEE Transactions on Computers*, vol. 69, no. 11, pp. 1654–1667, 2020.
- [3] M. Amin, D. Shehwar, A. Ullah et al., “A deep learning system for health care IoT and smartphone malware detection,” *Neural Computing and Applications*, vol. 31, no. 11, pp. 1–12, 2020.
- [4] H. Darabian, A. Dehghantanha, S. Hashemi et al., “An opcode-based technique for polymorphic Internet of Things malware detection,” *Concurrency Computational Practice Expert*, vol. 32, no. 6, 2019.
- [5] H. Kim, J. Kim, Y. Kim et al., “Improvement of malware detection and classification using API call sequence alignment and visualization,” *Cluster Comput*, vol. 22, no. 1, pp. 921–929, 2017.
- [6] N. Alshahwan, E. T. Barr, D. Clark, G. Danezis, and H. D. Menéndez, “Detecting malware with information complexity,” *Entropy*, vol. 22, no. 5, pp. 575–603, 2020.
- [7] R. Sihwail, K. Omar, K. Zainol Ariffin, and S. Al Afghani, “Malware detection approach based on artifacts in memory image and dynamic analysis,” *Applied Sciences*, vol. 9, no. 18, pp. 3680–3691, 2019.
- [8] D. Javaheri and M. Hosseinzadeh, “A framework for recognition and confronting of obfuscated malwares based on memory dumping and filter drivers,” *Wireless Personal Communications*, vol. 98, no. 1, pp. 119–137, 2018.
- [9] F. Xiao, Y. Sun, D. Du et al., “A novel malware classification method based on crucial behavior,” *Mathematical Problems in Engineering*, vol. 2020, no. 3, Article ID 6804290, 2020.
- [10] J. Zhang, Z. Qin, H. Yin, L. Ou, and K. Zhang, “A feature-hybrid malware variants detection using CNN based opcode embedding and BPNN based API embedding,” *Computers & Security*, vol. 84, no. 7, pp. 376–392, 2019.
- [11] F. O. Catak, A. F. Yazı, O. Elezaj et al., “Deep learning based sequential model for malware analysis using Windows exe API Calls,” *Peerj Computer Science*, vol. 6, no. 81, pp. 285–307, 2020.
- [12] L. Nataraj, S. Karthikeyan, G. Jacob et al., “Malware images: visualization and automatic classification,” in *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, pp. 401–407, ACM press, Pittsburgh, PA, USA, 2011.
- [13] Y. Dai, H. Li, Y. Qian, and X. Lu, “A malware classification method based on memory dump grayscale image,” *Digital Investigation*, vol. 27, no. 12, pp. 30–37, 2018.
- [14] S. Venkatraman, M. Alazab, and R. Vinayakumar, “A hybrid deep learning image-based analysis for effective malware detection,” *Journal of Information Security and Applications*, vol. 47, no. 8, pp. 377–389, 2019.
- [15] G. Daniel, M. Carles, P. Jordi et al., “Using convolutional neural networks for classification of malware represented as images,” *J. Comput. Virol. Hacking Tech*, vol. 15, no. 1, pp. 15–28, 2019.
- [16] V. Verma, S. K. Muttoo, and V. B. Singh, “Multiclass malware classification via first- and second-order texture statistics,” *Computers & Security*, vol. 97, Article ID 101895, 2020.
- [17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proceedings of Neural Information Processing System Foundation*, pp. 1097–1105, IEEE press, Lake Tahoe, NV, USA, 2012.
- [19] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, <http://arxiv.org/abs/1409.1556>.
- [20] K. He, X. Zhang, S. Ren et al., “Deep residual learning for image recognition,” 2016, <http://arxiv.org/abs/1512.03385>.
- [21] C. Szegedy, W. Liu, Y. Jia et al., “Going deeper with convolutions,” 2014, <http://arxiv.org/abs/1409.4842>.
- [22] G. Huang, Z. Liu, L. van der Maaten et al., “Densely connected convolutional networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, IEEE press, Honolulu, HI, USA, 2017.
- [23] Z. Huang, Y. Zhao, X. Li et al., “Application of innovative image processing methods and AdaBound-SE-DenseNet to optimize the diagnosis performance of meningiomas and gliomas,” *Biomedical Signal Processing and Control*, vol. 59, Article ID 101926, 2020.
- [24] X. Li, X. Shen, Y. Zhou et al., “Classification of breast cancer histopathological images using interleaved DenseNet with SENet (IDSNet),” *PLoS One*, vol. 15, no. 5, pp. 1–13, 2020.
- [25] W. Tong, W. Chen, W. Han, X. Li, and L. Wang, “Channel-attention-based denseNet network for remote sensing image scene classification,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, no. 4, pp. 4121–4132, 2020.
- [26] Q. Xie, K. Zhou, and X. Fan, “Feature attention based detection model for medical text,” *Journal of Intelligent & Fuzzy Systems*, vol. 37, no. 4, pp. 4585–4594, 2019.
- [27] L. Yang, P. Wang, H. Li, Z. Li, and Y. Zhang, “A holistic representation guided attention network for scene text recognition,” *Neurocomputing*, vol. 414, no. 1, pp. 67–75, 2020.
- [28] Y. Yang, X. Wang, Q. Zhao, and T. Sui, “Two-level attentions and grouping attention convolutional network for fine-grained image classification,” *Applied Sciences*, vol. 9, no. 9, pp. 1939–1953, 2019.
- [29] L. Mou and X. X. Zhu, “Learning to pay attention on spectral domain: a spectral attention module-based convolutional network for hyperspectral image classification,” *IEEE*

- Transactions on Geoscience and Remote Sensing*, vol. 58, no. 1, pp. 110–122, 2020.
- [30] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” 2018, <http://arxiv.org/abs/1709.01507>.
 - [31] H. Yakura, S. Shinozaki, R. Nishimura et al., “Neural malware analysis with attention mechanism,” *Computers & Security*, vol. 87, Article ID 101592, 2019.
 - [32] Z. Gao, Ji. Xie, Q. Wang et al., “Global second-order pooling convolutional networks,” 2019, <http://arxiv.org/abs/1811.12006v2>.
 - [33] S. Woo, J. Park, J. Y. Lee et al., “CBAM: convolutional block attention module,” 2018, <http://arxiv.org/abs/1807.06521v2>.
 - [34] Q. Wang, B. Wu, P. Zhu et al., “ECA-Net: efficient Channel attention for Deep Convolutional neural networks,” 2020, <http://arxiv.org/abs/1910.03151>.
 - [35] Z. Liu, J. Du, M. Wang, and S. S. Ge, “ADCM: attention dropout convolutional module,” *Neurocomputing*, vol. 394, no. 6, pp. 95–104, 2020.
 - [36] Y. Yang, C. Xu, F. Dong, and X. Wang, “A new multi-scale convolutional model based on multiple attention for image classification,” *Applied Sciences*, vol. 10, no. 1, pp. 101–118, 2019.
 - [37] W. Zeng and M. Li, “Crop leaf disease recognition based on self-attention convolutional neural network,” *Computers and Electronics in Agriculture*, vol. 172, no. 3, pp. 105341–105347, 2020.
 - [38] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 818–833, IEEE press, Washington, DC, 2014.
 - [39] S. Zagoruyko and N. Komodakis, “Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer,” 2017, <http://arxiv.org/abs/612.03928v3>.
 - [40] S. Jeon and J. Moon, “Malware-detection method with a convolutional recurrent neural network using opcode sequences,” *Information Sciences*, vol. 535, no. 5, pp. 1–15, 2020.
 - [41] A. Yewale and M. Sing, “Malware detection based on opcode frequency,” in *Proceedings of the 2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, pp. 646–649, IEEE press, Ram-anathapuram, India, 2016.
 - [42] Z. Cui, F. Xue, X. Cai, Y. Cao, G.-G Wang, and J. Chen, “Detection of malicious code variants based on deep learning,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3187–3196, 2018.
 - [43] X. Ma, S. Guo, H. Li et al., “How to make attention mechanisms more practical in malware classification,” *IEEE Access*, vol. 7, pp. 155270–155280, 2019.