processes

MDPI

Article

MalCaps: A Capsule Network Based Model for the Malware Classification

Xiaoliang Zhang *, Kehe Wu, Zuge Chen and Chenyi Zhang

School of Control and Computer Engineering, North China Electric Power University, No. 2 Beinong Road, Changping District, Beijing 102206, China; wkh@ncepu.edu.cn (K.W.); 1162127019@ncepu.edu.cn (Z.C.); 120192227088@ncepu.edu.cn (C.Z.)

* Correspondence: zhanghino@ncepu.edu.cn

Abstract: The research on malware detection enabled by deep learning has become a hot issue in the field of network security. The existing malware detection methods based on deep learning suffer from some issues, such as weak ability of deep feature extraction, relatively complex model, and insufficient ability of model generalization. Traditional deep learning architectures, such as convolutional neural networks (CNNs) variants, do not consider the spatial hierarchies between features, and lose some information on the precise position of a feature within the feature region, which is crucial for a malware file which has specific sections. In this paper, we draw on the idea of image classification in the field of computer vision and propose a novel malware detection method based on capsule network architecture with hyper-parameter optimized convolutional layers (MalCaps), which overcomes CNNs limitations by removing the need for a pooling layer and introduces capsule layers. Firstly, the malware is transformed into a grayscale image. Then, the dynamic routing-based capsule network is used to detect and classify the image. Without advanced feature extraction and with only a small number of labeled samples, the presented method is tested on an unbalanced Microsoft Malware Classification Challenge (MMCC) dataset and experimental results produce testing accuracy of 99.34%, improving on a number of traditional deep learning models posited in recent malware classification literature.

Keywords: capsule network; malware image; malware classification; hyper-parameter optimization; imbalanced dataset

Citation: Zhang, X.; Wu, K.; Chen, Z.; Zhang, C. MalCaps: A Capsule Network Based Model for the Malware Classification. *Processes* **2021**, *9*, 929. https://doi.org/10.3390/pr9060929

Academic Editors: Jie Zhang and Jae-Yoon Jung

Received: 13 April 2021
Accepted: 21 May 2021
Published: 25 May 2021

1. Introduction

Malware (a portmanteau for malicious software) is any software intentionally designed to cause damage to a computer, server, or computer network. It refers to the malicious program that is made or used by the attacker, spread by mobile storage media or network, and destroy the availability of information system and steal users' private information without authorization [1]. The criteria used to determine malicious code are: unauthorized and malicious. Malicious code includes: virus, snail, trojan horse, botnet, back door, rogue software, and other types of malicious programs. With the vigorous development of the Internet, malware has become one of the key threats. From the point of actual case in recent years, the outbreak of botnets, advanced persistent threat (APT), ransomware and other major network security incidents, malware act as the core component and cause substantial damages. In 2020, the AV-test system has detected more than 1.1 billion malicious codes as Figure 1 shows [2]. According to the 2020 China Internet Network Security Report, more than 62 million samples of computer malicious programs were captured throughout the year, and the average daily transmission times reached more than 8.24 million, involving more than 660,000 types of malicious programs, and including more than 731,000 ransomware viruses, among which the economic loss caused by Gandcrab virus alone reached 2 billion US dollars [3].

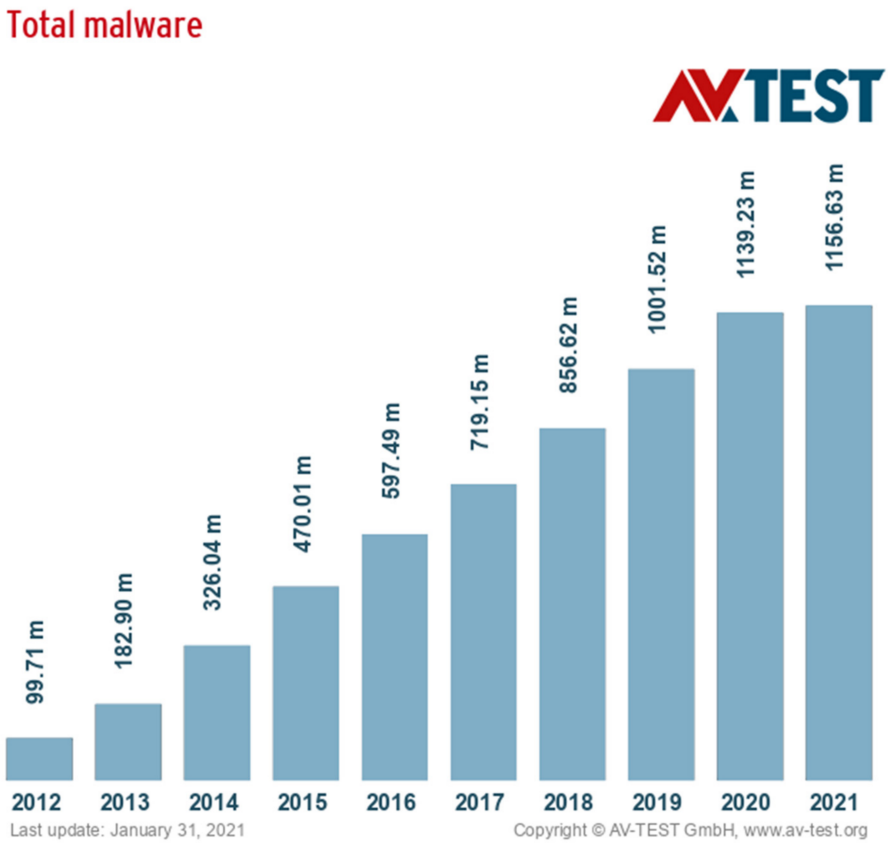


Figure 1. Overall development of new malware in the last 10 years, it can be seen that malwares is increasing prevalently. In 2021, there were more than 1.1 billion malicious codes as of the time this article was written. This figure is from av-test.org, accessed on 6 April 2021.

At present, malware is increasingly prevalent. In the process of fighting against malware samples, the analysis and detection technologies are also developing constantly. Since the 1990s, there have been researches on malware [4]; from rule matching and feature code extraction in the early stage, to dynamic and static detection and heuristic detection in the middle stage, and then to the current machine learning and multi-engine joint learning, malware detection technology has been developing continuously. However, the anti-detection technology is also constantly upgraded. Malware uses the technologies of shell, obfuscation, virtual machine protection, and so on to fight against various anti-killing technologies [5].

With the development of deep learning, it has replaced traditional methods to become a research hotspot of malware detection. Deep learning extracts the characteristics of malicious code from a large number of malware samples and classifies these features to obtain a malware recognition model, which has the obvious advantages of high automation and low resource consumption [6]. However, existing detection models based on deep learning still have some problems, such as weak ability of deep feature extraction, relatively complex model, and insufficient ability of model generalization, which need to be further explored and studied.

To address these limitations of existing approaches, we propose a completely new idea for malware detection based on capsule network in this paper. The capsule network, which is very different from other deep networks, was proposed by Hiton in 2017 [7]. Capsule networks are made up of capsules; a capsule is a set of neurons. Different from the traditional neural network, in a capsule, the set of neurons is vector or matrix. Each capsule represents various attributes of a specific entity in the image, such as the direction, position, and color of the object in the image, and the possibility of the entity's existence

is represented by the module length of the capsule vector. Low-level capsules transmit information to high-level capsules through dynamic routing mechanism in capsule network. Most prominently, the network can learn and store the spatial relationship information on the whole parts and the local of the object. Therefore, compared with the traditional neural network, capsule is a network with stronger feature extraction ability, especially for the extraction of details. Because of this ability, the number of samples needed for training a capsule network is much smaller than that of CNNs, and good results can be obtained without data enhancement.

Inspired by the works mentioned above, in this paper, we proposed a novel solution to detect malware based on the capsule network, without any features engineering. In this approach, the ideas and techniques of malware image visualization technology are borrowed. Using malware code image generation technology. We first convert the specially processed executable files into grayscale images. Next, the generated uniform images are sent into the capsule network to extract efficient features and train detection model. And then the trained capsule network model is used for the classification analysis of malware. The experiments are conducted on the Microsoft Malware Classification Challenge (MMCC) dataset to evaluate the proposed approach.

In conclusion, the main contributions of this work are summarized as follows:

1. Introduce malware image generation technology into the field of malware detection, which can completely store the feature information of malware;
2. Propose a new architecture in the domain of malware classification by implementing a novel modified Capsule Neural Network architecture;
3. Compare the Capsule Neural Network with convolutional neural network and demonstrates it's effectiveness in malware detection.

The remainder of this article is organized as follows: some related works are discussed in Section 2. Section 3 describes the processes of the proposed approach. In Section 4, the experiments and results are given to validate the scheme. Finally, in Section 5 we describe the conclusion and future work.

2. Related Work

In this section, prior works are introduced from two perspectives as follows:

1. traditional malware detection approaches;
2. shallow machine learning-based malware detection approaches;
3. deep learning-based malware detection approaches.

2.1. Traditional Malware Detection Approaches

The evolution of malware detection approaches was reviewed in detail by Caviglione et al. [8]. The literature showed five main approaches to malware detection: the early detection systems relied on signatures, the more recent ones use behavior-based, heuristic-based, energy-based, or bioinspired techniques. Signature-based methods rely on signatures, such as checksums, behavioral patterns, connections attributes, and metadata. They can match the malware content [9], protocols and payloads [10], being an effective method for detecting well-known malware, but often susceptible to code obfuscation techniques, such as packers, encryption, defor-mation and polymorphism technology, etc., and helpless against new malware. Behavior-based methods detect malware upon recognizing malicious or unwanted behaviors, which are compared with clean templates [11]. Common behaviors include modification of key registry of the system, and use of network communication resources, files, and mutex resources. There are many automated tools supporting behavior analysis such as Cuckoo [12], CWSandbox [13], and Ether [14]. Behavior-based methods can identify the type of malware and unrecognized malicious processes at runtime, so as to detect unknown malware. However, they are prone to false alarms and are time-intensive, resource-consuming, and can be evaded by mimicry attacks. Heuristic-based methods, including specification-based techniques, learn about the behaviors and characteristics of executable files, such as: API calls, byte N-grams, OpCodes, control

flow graphs, or their combinations [15,16]. They are able to detect known and unknown malware, however, they are time consuming, and often exhibit a low-level of accuracy. Energy-based methods exploit information about the energetic footprint of hardware and software as an indicator of ongoing attacks [17]. Bioinspired methods, such as Genetic Algorithms [18], Particle Swarm Optimization [19], and Ant Colony Optimization [20] are now used for malware features selection and optimization.

2.2. Shallow Machine Learning Based Malware Detection Approaches

Machine learning offers another way to effectively detect malware. Machine learning methods provide the ability to reduce the manual work required by traditional methods. It mainly includes four steps: data set construction, feature engineering, model training, and model evaluation [21]. Typically, features used to characterize malwares include the requested permissions, components, and filtering intent. Advanced static features such as n-grams features, control-flow diagrams [22], and API calling graphs obtained by decompiling malware files [23]. Dynamic behavior features, such as file operation, network operation, encryption operation, service opening, and system call, can be obtained by executing applications in an isolated environment. Combining these static and dynamic features to detect malicious software can achieve higher performance [24]. Models used for training include logistic regression [25], SVM [26], k-nearest neighbor (k-NN) [27], decision tree [28], random forest [29], and so on. The newest research on the machine learning-based malware detection focus on innovative feature spaces. Mohanasruthi et al., proposed an Application Program Interface Call Transition Matrix (API-CTM) [30], Farrokhmanesh et al., proposed to convert malware data bytes into audio signals [31]; however, the shortcoming is that the detection methods based on machine learning still rely on feature engineering and on complex or expert features to complete the learning task. The final effect of the machine learning based detection model is related to the selection of selected features, which is very subjective. Furthermore, these approaches are not conducive to efficient real-time malware classification monitoring. Therefore, models that can be automated are intuitively more useful and time-efficient in comparison.

2.3. Deep Learning Based Malware Detection Approaches

With the successful application of convolutional neural networks (CNNs) in ImageNet dataset, computer vision has entered a new era [32]. CNNs can classify images using raw pixel values without the need for complex feature engineering. In reference [33], the authors are the first to take the malware classification method as an image-based classification task. They represented the malware file as a binary string of zeros and ones. The vector is then reshaped into a matrix, and the malware file can be viewed as a grayscale image. The authors observed that after interpreting the original bytecode as grayscale images, images belonging to the same malware family looked very similar in layout and texture. In order to process this malware image as a classification task, all the malware executable binaries must first be reshaped as images. In this field, most of the previous work has used various CNNs structures to classify malware images [34,35]. Rezende et al., applied the ResNet-50 Deep CNN transfer learning to the MalIMG dataset by first converting each byte-graph to an RGB image. Compared with the KNN and GIST classification models, the classification accuracy of the proposed model achieved 98.62%, and without feature engineering [36]. Li et al., proposed a method combining self-attention mechanism with ResNet50 to detect malware variants. The method transformed malicious code into gray-scale image; then the data was augmented by overlapping. The method was tested on MalIMG dataset, and achieved 95.79% accuracy [37]. Priyadarsini et al., proposed a malware recognition procedure based on RNN-LSTM for the internet of things. Their model takes static, dynamic and hybrid highlights as inputs, which were selected using information gain algorithm. Compared with current KNN, LR, NB, DT, SVM, and DNN, the accuracy of their model achieved 98%, which is better than others [38]. Maryam et al., proposed a feature fusion method to combine the features extracted from pre-trained AlexNet and Inception-v3 deep neural

networks with features attained using segmentation-based fractal texture analysis (SFTA) of images representing the malware code. The presented method is evaluated on the Malimg dataset, achieving an accuracy of 99.3% [39]. Robertas et al., proposed an ensemble learning-based architecture and explored 14 machine learning algorithms as meta-learners, with neural networks used as base learners. They conducted experiments on a dataset that included malware and benign files from Windows with PE headers. The results showed that Extra Trees algorithm as a meta-learner and an ensemble of dense ANN and 1-D CNN models obtained the best accuracy value on their dataset [40].

Despite the impressive results of these papers, in terms of accuracy, the models can be further improved by adding additional layers and layers to extract more feature complexity. The first requirement for any CNN, regardless of the architecture, is a large amount of training data that is not readily available in the field of malware detection. Another limitation is the loss of information due to local details such as position and attitude during CNN pool operation, which may not have a significant impact on image processing, but this information may be critical to the malware data [41].

Through the above analysis, it is found that there are some problems to be solved in traditional malware detection algorithms, detection methods based on machine learning, and convolutional neural network:

1. the detection rate of traditional static detection algorithm is obviously reduced in the face of code obfuscation, shell, signature, and other camouflage technologies;
2. the dynamic detection method has higher requirements on the system resources;
3. feature extraction and screening based on machine learning is too complicated;
4. the convolutional neural network detection method loses information in the process of feature extraction and has a large demand on the data of training samples.

A study by Hinton et al. attempted to overcome the limitations of CNN. They depicted human visual perception as a deconstruction of images in the brain to simulate its hierarchy [42]. In 2017, Hinton proposed the capsule network model for the first time, and this model is considered to be an important neural network model of the next generation. Capsule networks are made up of capsules. A capsule is a set of neurons, different from the traditional neural network, in a capsule, the set of neurons is a vector or matrix. Each neuron represents various attributes of a specific entity in the image, such as the direction, position, and color of the object, and the possibility of the entity's existence is represented by the module length of the capsule vector. Capsule network is a network in which low-level capsules transmit information to high-level capsules through dynamic routing mechanism [7,42]. Therefore, capsule is a network with stronger feature extraction ability, especially for the extraction of details, compared with the traditional neural network. Due to its excellent performance, the capsule network model has been well applied to small sample image classification tasks, which contains only several thousand samples, such as text classification [43], visual reconstruction [44], brain tumor classification [45], diagnosis of rotator cuff tears, capsule GAN [46], and deep reinforcement learning [47]. As mentioned in relevant literature, capsule network shows potential application in various fields.

As for the cybersecurity field, there are only very few papers in literatures about capsule networks within the malware classification domain. Cayir et al., propose a Random CapsNet for imbalanced malware based on bootstrap aggregating methods [48]. Their RNCF on the dataset achieves a 99.56% accuracy, though an impressive accuracy result, there are arguably two limitations to this paper. The first limitation is the number estimators in the RCNF model, in their implementation, an RCNF model can only contain up to 10 capsules. The second limitation is the training time, training of an RCNF with 10 capsules for the MMCC dataset takes five hours. This is because their models are more complex and require more parameters. Wang et al. proposed a novel malware detection and classification method based on capsule network [49]. Their capsule network structure only contains one layer of convolution layer. Phaye et al., argued that feature maps learned by the first convolution layer in the baseline CapsNet model only learns basic features, which may lack flexibility for deeper feature extraction [50]. Furthermore, they only did the

binary classification detection, and the model was not verified in the public test set. This paper will discuss our proposed Malcaps, and the role and significance of this technology.

3. Method Descriptions

In this section, we describe our capsule network-based approach for malware classification in detail, including the dataset we used for our experiments, malware data preprocessing, capsule neural network architectures, and our proposed capsule network. The overall flow of the proposed methodology is shown in Figure 2.

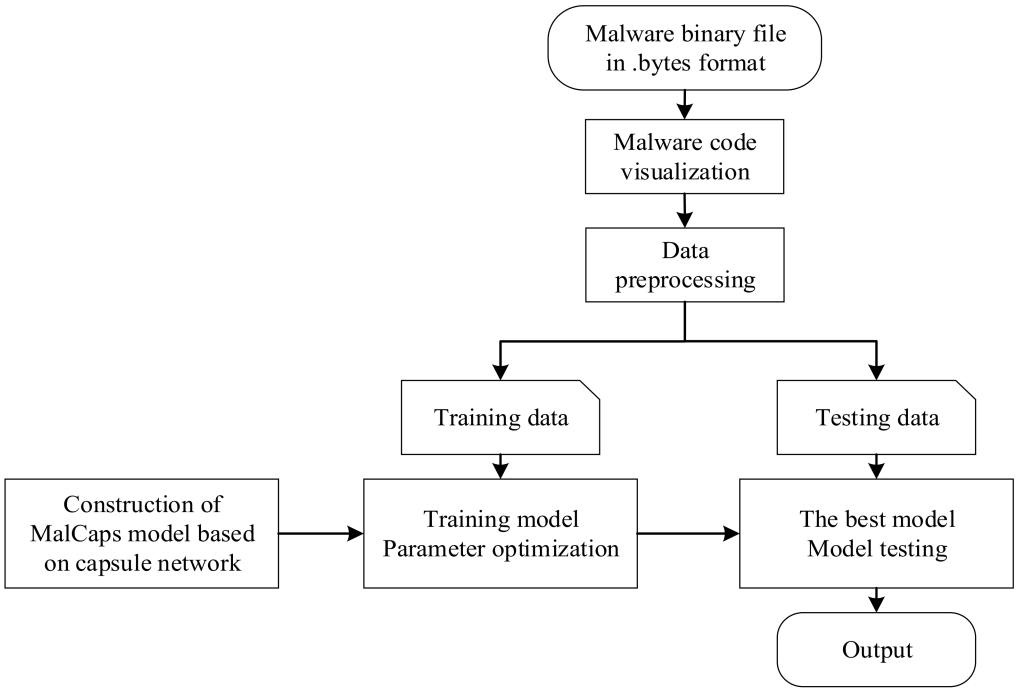


Figure 2. Flow diagram of the proposed framework.

3.1. Microsoft Malware Classification Challenge Dataset

The malware dataset used in this paper is from a Microsoft project on Kaggle, Microsoft Malware Classification Challenge (MMCC) [51]. The dataset has become the standard benchmark to evaluate machine learning techniques for the task of malware classification. It consists of a set of known malware files representing a mix of 9 different families: (1) Ramnit, (2) Lollipop, (3) Kelihos_ver3, (4) Vundo, (5) Simda, (6) Tracur, (7) Kelihos_ver1, (8) Obfuscator.ACY, (9) Gatak. Each sample identified by an identifier, a 20-character hash value uniquely and its class label, has a corresponding assembly file and a raw hexadecimal representation of the file's binary content, without the PE header (to ensure sterility). The dataset has 21,741 samples, with 10,868 for training and the other 10,873 for testing, being a dataset of almost half a terabyte uncompressed. Table 1 shows the distribution of various classified categories present in the training dataset. As the focus of this paper is the application of a novel capsule network-based technique to classify malware based on the raw binary file content, we only consider the raw hexadecimal file representations, and convert it to its binary representation.

Table 1. Malware families in the Microsoft Malware Classification Challenge dataset for Kaggle.

Class No.	Family Name	Train Samples	Malware Type
0	Gatak	1013	Backdoor
1	Kelihos_ver1	398	Backdoor
2	Kelihos_ver3	2942	Backdoor
3	Lollipop	2478	Adware
4	Obfuscator.ACY	1228	Any kind of obfuscated malware
5	Ramnit	1541	Worm
6	Simda	42	Backdoor
7	Tracur	751	TrojanDownloader
8	Vundo	475	Trojan

3.2. Transforming Malware Byte Files into Images

At present, there is lots of research on the methods of malicious code visualization. Nataraj et al. [33] first proposed to visualize malware binaries as grayscale images and classify malware according to the similarity of image textures. Some researchers refer to the processing methods of images of different sizes in the image field, and use deformation processing such as cutting or scaling to convert malicious code images to a fixed size [52]; however, these methods will cause the loss of malicious code data and the destruction of code structure. This paper utilized the method to visualize malicious code, which converts binary code into three-channel image as Wang et al. did in their paper [49]. Since three-channel image (24 bit pixels per sample) can hold 16,777,216 characteristics, single-channel image only has 256 characteristics (8 bit pixels per sample). Three-channel images have stronger feature representation capability. For a given malicious code binary, it reads three 8-bit binary numbers into decimal integers (in the range 0 to 255), reshapes those integers into vectors, and finally generates a two-dimensional array whose width and height vary depending on the size of the file. Finally, the array is visualized as a three-channel image. In order not to affect the final analysis results, 0 is used to fill in the case that the content read in the last time is less than 24 bits. The mapped image is saved as an uncompressed PNG image. The process of mapping malicious code to images is shown in Figure 3.

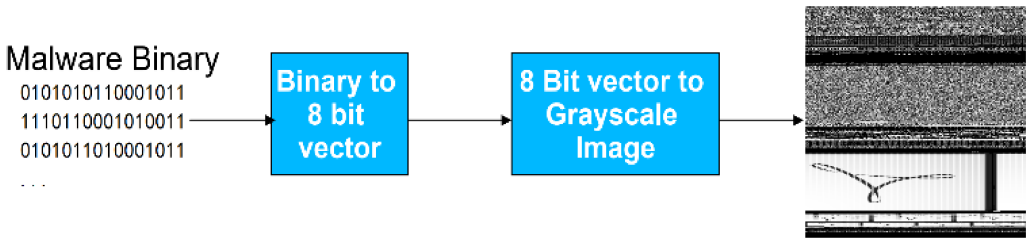


Figure 3. The process of visualizing malware as an image. Malware byte files are converted into 8-bit vectors, and then reshaped to two-dimensional arrays, at last saved as three-channel images.

Figure 4 is an image of some sample samples from the MMCC dataset. It is clear from the figure that image samples from different software families have different characteristic texture features, and samples from the same malware family are very similar. This is because a large amount of malicious code is now using code reuse technology, some key code blocks are reused, so the same code often contains the same module; code is similar, different code is different. The image texture features can reflect the similarity and difference effectively. Therefore, this method will provide effective sample inputs for capsule network-based malware classification.

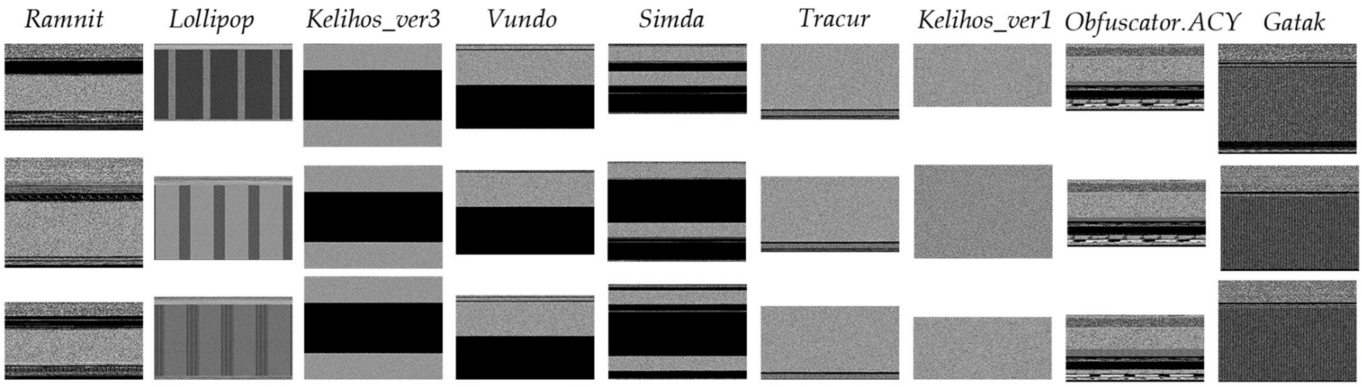


Figure 4. Image created from the BYTE files of the malware samples in MMCC dataset. It can be seen that images from different software families have distinct texture features and the same malware family samples are very similar.

3.3. Capsule Neural Networks

Capsule Neural Network is the latest development in the field of deep learning architectures, introduced by Hinton et al. in 2017 [7] to overcome limitations from the traditional CNN approach. Capsule is a carrier that contains a group of organized neurons, each of which represents various properties of a particular entity that appears in the image. These properties can include many different types of instantiation parameters, such as attitude (position, size, direction, deformation, speed, hue, texture, and so on). A very special property in the capsule is the presence of an instance of a category in the image. Its output value is the probability of the existence of the entity. The capsule network is designed to mimic the way our brains process vision, called reverse rendering. Humans solve an image into many hierarchical subparts, and construct the relationships between these subparts in order to recognize familiar objects; that is the main way the capsule tries to mimic. The capsule is placed in each part of the image, and the capsule output indicates whether the subparts of the image are located in that position.

The capsule network consists of three kinds of hidden layers: convolutional layer, primary capsule layer, and digit capsule layer. The convolution layer extracts features from images through a convolutional filter of 9×9, the number of channels is 256, and the step length is 1, which is activated by ReLU function, and create a 20×20 local feature map. The primary capsules convert the feature map from scalars into the vectors. A total of 32 distinct 6×6 capsules in 8 dimensions transform the scalar quantities into vectors with directional information. The digit capsule layer outputs the capsule length of 16. These capsules are calculated from the capsules of the primary capsule layer through dynamic routing algorithm, known as routing-by-agreement [7].

The general calculation process of capsule network is shown as follows:

1. $u_i \in R^{k\times 1}, i = 1, 2, \ldots, n$, is the input of lower capsule, where n represents the number of capsules, k represents the number of neurons in each capsule (vector length).
2. Apply a transformation matrix (or a matrix multiplication, $W_{ij} \in R^{p\times k}$, to the lower layerto the lower layer's vector which represents important spatial information between low-level and high-level features, p represents the number of neurons in the output capsule, and converts the input $u_i \in R^{k\times 1}$ into a "prediction vector" $\hat{u}_{j|i} \in R^{p\times 1}$:

$$\hat{u}_{j|i} = W_{ij} u_i \tag{1}$$

   The weighting matrix W_{ij} is learned during the back propagation procedure.
3. Then the weighted sum of all the obtained prediction vectors is carried out:

$$s_j = \sum_i c_{ij} \hat{u}_{j|i} \tag{2}$$

where s_j is called the total input of higher capsule, c_{ij} is the coupling coefficient calculated by dynamic routing, and

$$\sum_j c_{ij} = 1 \tag{3}$$

conceptually, the c_{ij} represents the probability distribution of the capsule i activating the capsule j.

4. Finally, the output of higher capsule j is obtained by activating a non-linear "squashing" function, it is used to transform short vectors to almost zero length and long vectors close to 1, and the direction of the vector remains the same.

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|} \tag{4}$$

The coupling coefficient c_{ij} is calculated by the softmax function as equation:

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \tag{5}$$

where b_{ij} represents the degree of correlation between capsules in layer L and layer L+1, and the initial value of b_{ij} is 0. Update b_{ij} through Equation (4) until the iteration requirements are met.

$$b_{ij} = b_{ij} + \hat{u}(j|i) \cdot v \tag{6}$$

The calculation of the capsule network loss function is the marginal loss of a specific class of object c, which can be calculated by Formula (7). On the basis of experiments, the ideal parameters are set as $m^+ = 0.9$ and $m^- = 0.1$, and remain unchanged in this paper. The lambda coefficient λ prevents initial learning from reducing the size of the class's activity vector and is set to 0.5. The total loss is the sum of the total losses of all categories. If there is an object C of a particular class, then T_c =1.

$$L_c = T_c \max(0, m^+ - \|v_c\|)^2 + \lambda(1 - T_c)\max(0, \|v_c\| - m^-)^2 \tag{7}$$

The original paper sets the ideal parameters based on experiments as $m^+ = 0.9$ and $m^- = 0.1$ and are kept as constant for this paper. The lambda coefficient λ prevents initial learning from reducing the size of activity vectors for classes and is set at 0.5. The total loss sums the total loss of all classes. If an object of a specific class, c, is present then $T_c = 1$ [53].

One of the primary justifications to why a CapsNet theoretically should perform better than CNN specifically for the task of malware classification, is that malware binaries represented as images have clear sections. In other words, the proposed model should allow for better hierarchical relationships, which is important in a dataset that is comprised of images with set sections.

3.4. Our Proposed Capsule Network

Convolutional neural network can extract image feature information very well, while capsule network has a better ability to identify the spatial position relationship, direction, and other attributes of the image, and the transmission loss of image feature information is very small. In this paper, we propose a modified capsule network architecture (MalCaps) for malware classification based on malware visualization and capsule network.

Two strategies were used to increase the accuracy of the proposed MalCaps. Firstly, the architecture proposed in this paper increases the number of convolution layers and tries to improve the learning of discriminative features. On the basis of experiments, the proposed model chooses to increase the number of convolution layers to two, which could also improve accuracy by creating more complex features before feeding to the primary capsule layer. As suggested in the literature, the more layers of the model, the more features

the model can extract. Mathematically speaking, the structure of capsule neural network is much more complex, because it can input and output vectors rather than scalars. The advantage of not adding more layers is that the training time should be minimized.

Additionally, a Random Search is used to run through the number of neurons in the first convolutional layer as well as the second convolutional layer to improve the CNN architecture, the neuron number, the optimal epoch number, and the optimal batch size. There are many algorithms that can iterate various experiments with different hyper-parameter configurations. Through their empirical experiments, Bergstra and Bengio showed that random search and test were more effective than grid search or manual search in optimizing hyper-parameters [38].

The network architecture of our proposed MalCaps model consists of the convolutional part, the primary capsule part, and the classification part, as Figure 5 shows, and the calculation process is shown in Formula (8). The input of the network is a $64 \times 64 \times 3$ grayscale image, which is converted from malware training samples according to the method in previous description. A down sampling method is used to unify the original image of fixed format as the input of the model. The convolutional part utilizes 2 convolutional layers instead of 1 for feature extraction. The extra convolutional layers can provide more specified feature maps that can be fed into the primary capsule. The first convolutional layer of MalCaps contains 5 kernels of size 3×3 with a stride of 1, without padding, using the ReLU function as a nonlinear activation function. Output of the first convolution layer was given to the second convolution layer as input to extract useful low-level features. The second convolutional layer retains the same structure as the first. The next layer is a primary capsule layer, which contains 32 different capsules that systematizes the processed feature maps into capsule nodes. Each capsule node contains 3×3 convolutional kernels at a stride of 3, generates 32 distinct 20×20 capsules in 8 dimensions and fed into classification capsule. The final layer consists of a high-level capsule layer, referred as "Class Capsules", that performs the classification. The routing iteration is 3 in the capsule layer and the dimension of each capsule is 8, as suggested to be optimal in [7]. Finally, 9×16 (9 represents the number of categories, and 16 is the dimension of each capsule) vector features output is normalized by L2 normalization as the output of the network, which represents the probability of the existence of each malware category. The decoder unit consists of three fully connected layers with 512, 1024, and 12,288 neurons, respectively. The number of neurons in the last fully connected layer is the same as the number of pixels in the input layer, because the goal is to minimize the sum of the squared differences between the input image and the reconstructed image.

$$
\begin{aligned}
&\text{Input} -> \text{Conv1} \quad [64,64,3] \xrightarrow{5\ conv(3\times3)} [62,62,5] \\
&\text{Conv1} -> \text{Conv2} \quad [62,62,5] \xrightarrow{5\ conv(3\times3)} [60,60,5] \\
&\text{Conv2} -> \text{PrimaryCaps} \ [60,60,5] \xrightarrow{8\times32\ conv(3\times3)} [20,20,8\times32] \xrightarrow{reshape} [12800,8] \xrightarrow{activation(squash)} [12800,8](u_i \in R^8(i=1,2,\ldots,12800)) \\
&\text{PrimaryCaps} -> \text{MalCaps}\ u_{\mathrm{i}} \in R^8 \xrightarrow{W_{ij}(j=0,1,\ldots,8)} \hat{u}_{j|i} = W_{ij}u_i \underbrace{\xrightarrow{sum} s_j = \sum_i c_{ij}\hat{u}_{j|i} \xrightarrow{squash} v_j = squash(s_j) \in R^{16}}_{DynamicRouting} \\
&\text{Output} \quad v_j \xrightarrow{L2} (\|v_0\|, \|v_1\|, \ldots, \|v_8\|)
\end{aligned}
\tag{8}
$$

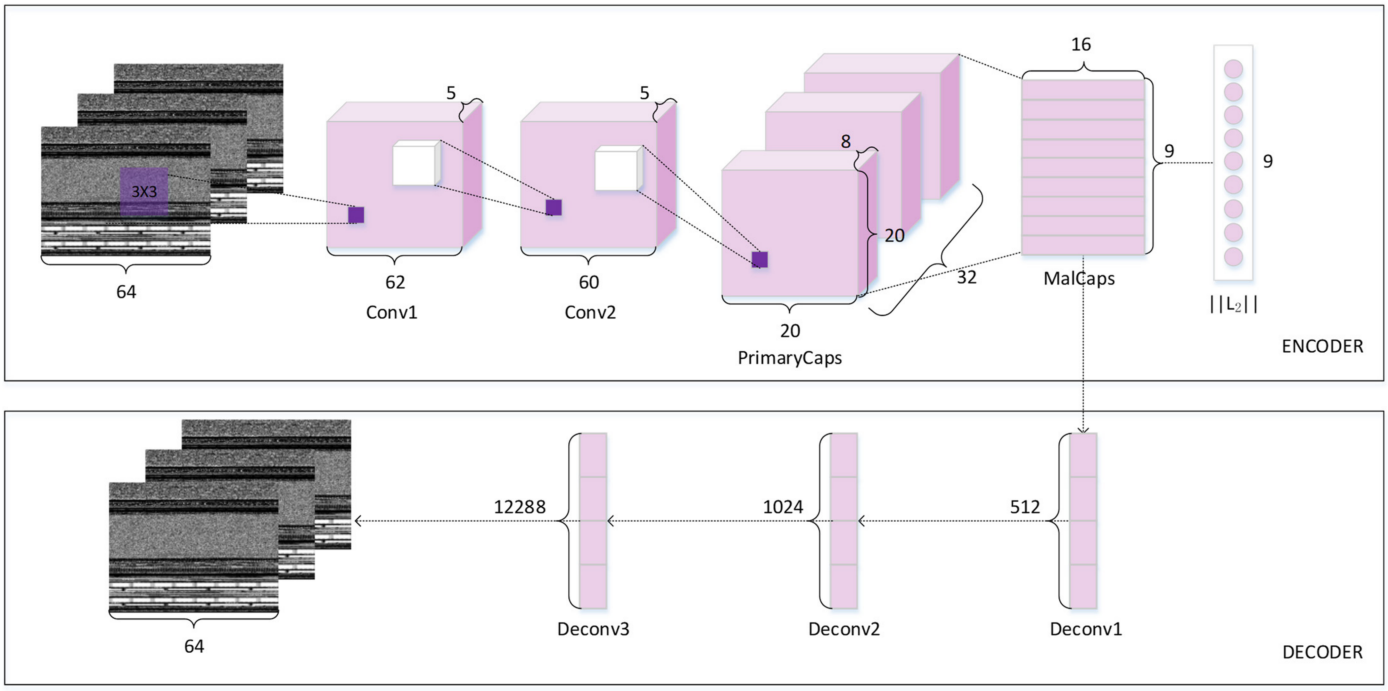


Figure 5. Architecture of the proposed MalCaps.

Other differences to the original paper is that the implementation uses a decay factor = 0.9 (for the learning rate decay) and step = 1 for the epoch, as shown to be optimal rates in experiments run by Guo [54].

4. Experiments

4.1. Evaluation Metrics

In this section, we verify the validity of the proposed method on Microsoft Malware Classification Challenge dataset. Metrics including precision, accuracy, recall, and F1-Score are used for the quantitative assessment of performance as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{9}$$

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

$$\text{Accuracy} = \frac{TP}{TP + FP + TN + FN} \tag{11}$$

$$\text{F1} = \frac{2TP}{2TP + FN + FP} \tag{12}$$

where, TP is the true positives, FP is the false positives, TN is the true negatives, FN is the false negatives.

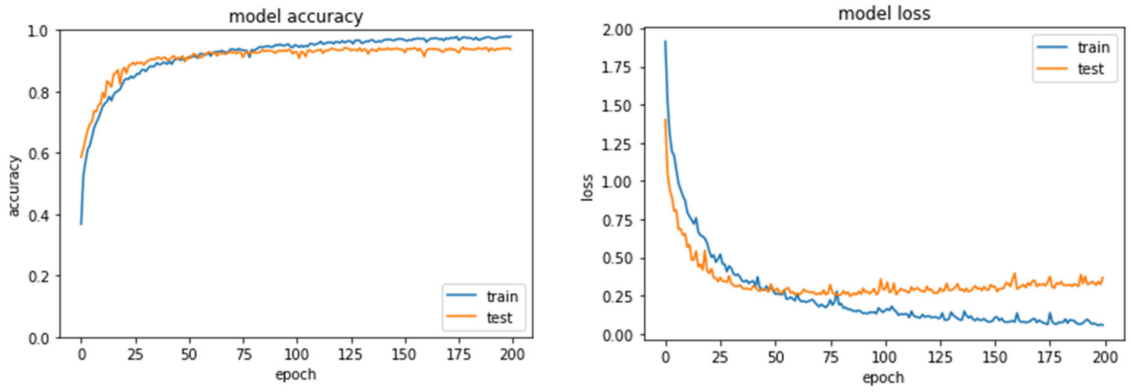
4.2. Results

Our experiments were implemented on a server with 2 Intel Xeon E5-2680 processors, with no GPU acceleration, and 64 GB of RAM.

Malware images are divided into two parts for training and testing after image preprocessing. The MMCC dataset was split into two, 70% for training and 30% for testing, respectively. From Table 1 we can see the distribution of the classes in the dataset is highly imbalanced, with the number of samples per class ranging from 42 samples for the

class Simda to 2942 samples for the classKelihos_v3, which make this task an imbalanced classification. In order to solve the impact of data imbalance on model accuracy, the "class weight" parameter in the scikit-learn library is utilized to give higher weight to minority classes and lower weight to majority classes.

The progress of CNNs in the image recognition field has been remarkable. For performance comparison, we first conducted popular CNN architectures Le-net5 using the database, which resulted in the model's evaluating accuracy of 93.71%. The model accuracy and loss curve of training and testing as Figure 6 shows. It can be found that the training tends to be stable after approximately 25 epochs, but the model loss was slightly higher in the testing phase.



(**a**) (**b**)

Figure 6. Accuracy Curve (**a**) and Loss Curve (**b**) of Le-net5 model trained on the MMCC dataset.

Table 2 shows the training accuracy, precision, recall, F1, and MCC for each malware class. The statistics presented in Table 2 shows some promising result of the Le-net5 model, despite the simplicity of the model. The per-class precision shows most of malwares were correctly classified, with class 0, 6, 7, and 8 below 90%. Per-class recall shows that for class 6, just 28.57% were calculated correctly, that's because its proportion in the dataset is much smaller than the others. The per-class accuracy scores show all of the 9 classes were above 98%.

Table 2. Results of the training of Le-net5 on MMCC dataset.

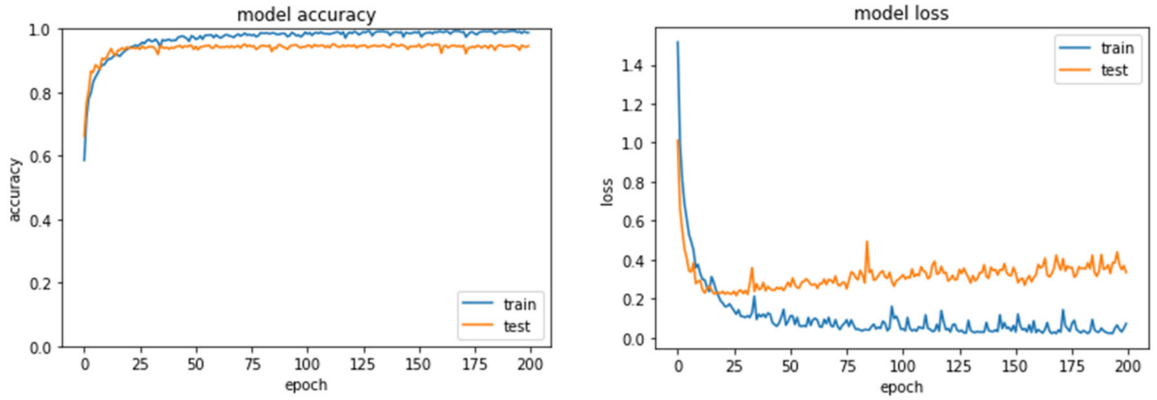
Class No.	Precision	Recall	Accuracy	F1
0	0.7866	0.9307	0.9701	0.8526
1	0.9888	0.9167	0.9959	0.9514
2	0.9847	0.9949	0.9945	0.9898
3	0.9835	0.9246	0.9784	0.9531
4	0.9174	0.8889	0.9802	0.9029
5	0.9241	0.8933	0.9752	0.9085
6	0.6667	0.2857	0.9972	0.4000
7	0.8561	0.8500	0.9811	0.8530
8	0.8264	0.9615	0.9885	0.8889

To optimize the hyper-parameters of the convolutional layers with in the proposed MalCaps, a random-search method was then conducted the compared Le-net5 model. In every training epoch, different hyper-parameter configurations were randomly searched. As Table 3 shows, the best accuracy, 99.03%, was got with the hyper-parameter configuration as: convolutional layer1 neuron1 = 5, convolutional layer neuron2 = 5, epoch = 50, batch = 100, which will be used in our proposed MalCaps.

Table 3. Optimization of convolutional layers' hyper-parameters for the Le-net5.

neurons1	neurons2	epochs	batch_size	Accuracy
25	100	200	1000	0.9436
100	25	200	1000	0.9655
20	50	150	200	0.9592
20	25	200	100	0.9770
20	10	100	20	0.9764
100	30	50	500	0.9776
25	10	100	50	0.9839
5	100	100	20	0.9850
100	20	200	500	0.9799
5	5	100	50	0.9903

The hyper-parameters optimized Le-net5 was trained on the MMCC dataset, and the summary statistics in Table 4 show some better and more realistic results as improvements to some accuracy results of per-class. The best accuracy was on class 7 with 99.86% accuracy, with 100% per-class precision. The average accuracy fluctuated between 95.79% and 98.7%, while during the test phase between 92.27% and 94.85%, as training accuracy showed in Figure 7.



(**a**) (**b**)

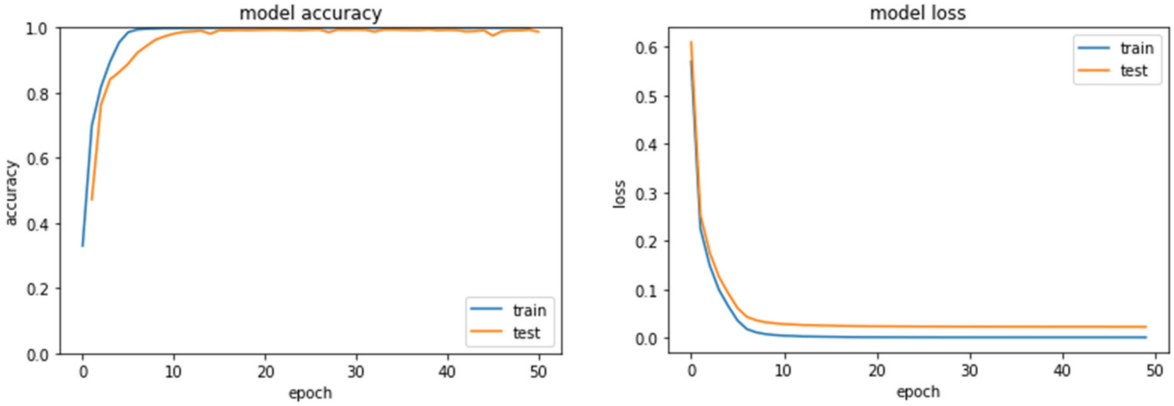
Figure 7. Accuracy Curve (**a**) and Loss Curve (**b**) of the hyper-parameters optimize Le-net5 model trained on the MMCC dataset.

Table 4. Results of the training of Le-net5 on MMCC dataset, where the hyper-parameters of convolutional layers were optimized.

Class No.	Precision	Recall	Accuracy	F1
0	0.8263	0.9653	0.9779	0.8904
1	0.9889	0.9271	0.9963	0.9570
2	0.9949	0.9966	0.9977	0.9957
3	0.9743	0.9516	0.9825	0.9628
4	0.9657	0.8756	0.9839	0.9184
5	0.9331	0.9300	0.9811	0.9316
6	1.0000	0.5714	0.9986	0.7273
7	0.8897	0.8643	0.9844	0.8768
8	0.8707	0.9712	0.9917	0.9182

The success of our MalCaps model was compared to that of CNN for the discrimination of malware based on the hyper-parameters optimized Le-net5. The MalCaps model was iterated over 50 epochs, Figure 8 shows training loss decreases over epoch iterations, whilst training accuracy increases, and resulting in an accuracy of 99.89%. These promis-

ing results suggest that this MalCaps can be used in malware classification. This final implementation achieved an overall precision of 99.34%, which was better than that of the hyper-parameter optimized CNN model in the previous experiment.



(**a**) (**b**)

Figure 8. Accuracy Curve (**a**) and Loss Curve (**b**) of our proposed MalCaps model trained on the MMCC dataset.

Finally, we present the confusion matrix of our proposed model, showed in Figure 9, which reveals that Malcaps is capable of classifying many malware families correctly. But it has problems mainly in classifying samples from Kelihos_ver1, Lollipop, Obfuscator.ACY, Ramnit, and tracur and they misclassified some samples as belonging to the Gatak malware's family. In particular, the major misclassifications are produced from samples of the Lollipop family.

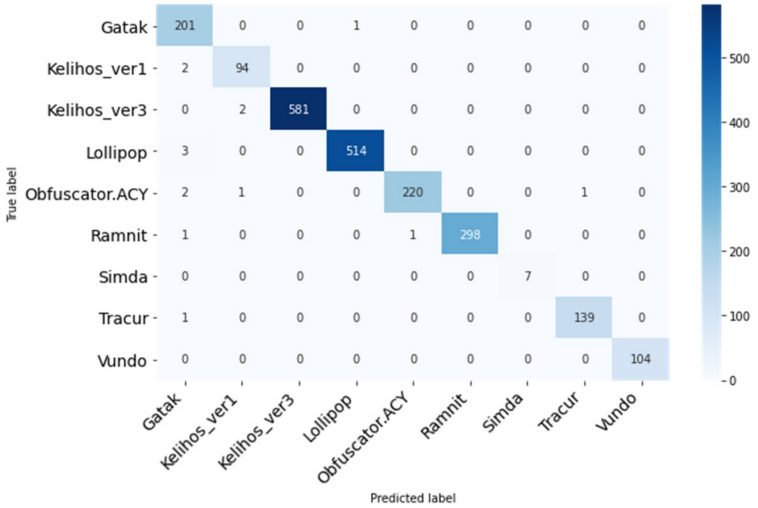


Figure 9. Confusion matrix of our MalCaps classification results.

There are a number of alternative deep learning architectures that exist in the field of malware classification. To further evaluate the performance of our proposed approach, we compared MalCaps with state-of-the-art methods in the literature that have evaluated their models on the dataset provided for the Kaggle's Microsoft Malware Classification Challenge, which depending on the feature of grayscale image representing the malware's binary content or a set of features extracted from it using any feature extractor technique, used as input for the training algorithms. The results are shown in Table 5.

Table 5. Comparisons of different networks on the MMCC dataset depending on the feature of grayscale image.

Approach	Feature Type	Classification Algorithm	Accuracy	F1
Ahmadi et al.,2016 [55]	Haralick features	XGBoost	0.9690	0.9282
Ahmadi et al.,2016 [55]	Local Binary Pattern features	XGBoost	0.9724	0.9530
Narayanan et al., 2016 [56]	PCA features	1-NN	0.9660	0.9102
Fu, 2018 [57]	Gray-image	KNN-3	0.931	0.920
Cui, 2018 [58]	Gray-image	OtherCNN	0.945	0.945
Gibert et al., 2019 [59]	128 × 128 Grayscale Image	CNN	0.9750	0.9400
Narayanan et al., 2020 [60]	Grayscale Image and Opcodes	Ensemble Approach using SVM	0.9980	-
Çayır et al., 2020 [48]	Images from BYTE and ASM	RCNF Ensemble Model	0.9872	0.9661
Roseline et al., 2020 [61]	Grayscale Image	Deep Forest Model	0.9720	0.9720
MalCaps	Three channel Image	Capsule Neural Network	0.9934	0.9443

These approaches include traditional classification approaches [55–57], CNN [58,59], ensemble model [48,60], deep forest model [61], and our proposed approaches. As observed in Table 5, MalCaps achieved comparable results to others, and achieved a higher detection rate and macro F1-score. Thus, we demonstrate that capsule network based mothed can be successfully complemented with grayscale image features to achieve good results in the malware classification task. In addition to these, Narayanan et al. reported accuracy scores for the MMCC dataset, their accuracy is the highest one for the dataset, but their network has two complex phases and they do not use F1-Score despite the dataset is highly imbalanced.

5. Discussion

Despite our MalCaps achieving a relatively commendable accuracy rate, these results are not without limitations. The per-class accuracy varies and the confusion matrices presented in the previous section show that the models failed to classify some of the family classes that were a minority class. Additionally, the time used to train the model is higher, the optimized CNN model took around 24 min trained in 200 epochs compared to a capsule neural network of 289 min trained in 50 epochs.

Capsule Neural Networks are the latest development in the field of deep learning, but a lot of research should be developed before they can be used into industry to protect against malware. For future work, there are a number of approaches to improve the proposed MalCaps, in ways of model effectiveness, efficiency, and time taken. From the view of algorithm, a more efficient way is to increase the depth of networks like CNNs. Nevertheless, the current trade-offs are that in order to train such a deep capsule network, much more computational power is required. This may not be the most ideal way, for that it is assumed that the mapping learned from the data will be valid in the future. However, this assumption is not valid in the malware domain. Malwares evolve over time, new malwares are being created daily, and training may not currently be fast enough to keep up with malware development. Additionally, the similarity between previous and future versions will degrade slowly over time; this is known as the problem of concept drift. From the view of data, take more information of malware as input will be helped. One approach is to process the images in color, rather than greyscale, which allows more feature details, such as character information, byte stream information, PE structure information, and so on, which are encoded in RGB channel. These features, which represents the malware byte files, are not available in purely a greyscale image. Another approach is to preserve the semantic meaning of each byte in the raw binary file in the preprocessing step, though this approach means we need a suitable way to compress a large binary file without losing the semantic meaning in the final representation.

6. Conclusions

Distinguishing and classifying different types of malware is an important task as it provides information to better understand how the malware has infected the computers or devices, their threat level, and how to protect against them. In this paper, we proposed a modified capsule network, MalCaps, for the classification of malwares. This method first converts the software samples into greyscale images and then performs the capsule network model training. To the best of our knowledge, at present there are only a few studies using capsule network in malware analysis, probably because this is a very recent technique in the deep learning field. Experiments show that the proposed MalCaps architecture was found to be successful and the results were quite similar to those of an expert, achieving the highest accuracy of 99.37%. One of the key benefits of the capsule network model over traditional deep learning models (e.g., CNNs) is that it takes into account the spatial information which is usually lost in a pooling layer of CNNs.

This paper describes an attempt to apply the capsule network in the field of malware classification. Capsule network may be considered one of the most innovative and successful models in the field and should be evaluated in further studies. For the future, if the appropriate computational power is made available, deeper capsule network architectures are discussed as potential solutions to increase generalization performance as well as different data pre-processing techniques. Furthermore, real-time data will be trained using this method.

Author Contributions: Conceptualization, X.Z. and K.W.; data curation, X.Z. and Z.C.; funding acquisition, X.Z.; methodology, X.Z.; software, X.Z. and C.Z.; writing–original draft, X.Z.; writing–review & editing, X.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the Science and Technology Project of the State Grid Corporation of China, Grant number 5700-202024193A-0-0-00 and in part by the Fundamental Research Funds for the Central Universities, Grant number 2020MS082.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found at: https://www.kaggle.com/c/malware-classification/data, accessed on 12 April 2021.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liu, L.; Wang, B.-S.; Yu, B.; Zhong, Q.-X. Automatic malware classification and new malware detection using machine learning. Front. *Inf. Technol. Electron. Eng.* **2017**, *18*, 1336–1347. [CrossRef]
2. Available online: https://www.av-test.org/en/statistics/malware/ (accessed on 6 April 2021).
3. National Computer Network Emergency Response Technical Team/Coordination Center of China. 2019 China Internet Cybersecurity Report. Available online: http://www.cac.gov.cn/2020-08/11/c_1598702053181221.htm (accessed on 6 April 2021).
4. Denkins, P.; LeGrand, S. A model for the protection of reusable software. *ACM SIGSAC Rev.* **1991**, *9*, 3–18. [CrossRef]
5. Weiping, W. Researeh on Mechanism and Defense of Malicious Code. Ph.D. Thesis, The Chinese Academy of Science, Guangzhou, China, 2005.
6. Kolosnjaji, B.; Zarras, A.; Webster, G.; Eckert, C. *Deep Learning for Classification of Malware System Call Sequences*; Springer: Cham, Switzerland, 2016.
7. Sabour, S.; Frosst, N.; Hinton, G.E. Dynamic Routing Between Capsules. *Adv. Neur.* **2017**, *30*.
8. Caviglione, L.; Choras, M.; Corona, I.; Janicki, A.; Mazurczyk, W.; Pawlicki, M.; Wasielewska, K. Tight Arms Race: Overview of Current Malware Threats and Trends in Their Detection. *IEEE Access* **2021**, *9*, 5371–5396. [CrossRef]
9. Griffin, K.; Schneider, S.; Hu, X.; Chiueh, T.C. Automatic Generation of String Signatures for Malware Detection. In *Recent Advances in Intrusion Detection, Proceedings*; Kirda, E., Jha, S., Balzarotti, D., Eds.; Springer: Berlin, German, 2009; pp. 101–120.
10. Rafique, M.Z.; Caballero, J. FIRMA: Malware Clustering and Network Signature Generation with Mixed Network Behaviors. In *Research in Attacks, Intrusions, and Defenses*; Stolfo, S.J., Stavrou, A., Wright, C.V., Eds.; Springer: Berlin, German, 2013; pp. 144–163.
11. Aslan, O.; Samet, R. A Comprehensive Review on Malware Detection Approaches. *IEEE Access* **2020**, *8*, 6249–6271. [CrossRef]

12. Jamalpur, Y.S.N.S.; Raja, P.; Tagore, G.; Rao, G.R.K. Dynamic Malware Analysis Using Cuckoo Sandbox. In Proceedings of the 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, 20–21 April 2018.
13. Willems, G.H.; Holz, T.; Freiling, F. Toward automated dynamic malware analysis using cwsandbox. *IEEE Secur. Priv.* **2007**, *5*, 32–39. [CrossRef]
14. Dinaburg, A.; Paul, R.; Shari, M.; Lee, W. Ether: Malware analysis via hardware virtualization extensions. In Proceedings of the 15th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 27–31 October 2008.
15. Kozachok, A.V.; Kozachok, V.I. Construction and evaluation of the new heuristic malware detection mechanism based on executable files static analysis. *J. Comput. Virol. Hacking Tech.* **2017**. [CrossRef]
16. Alkhateeb, E.M.; Stamp, M. A Dynamic Heuristic Method for Detecting Packed Malware Using Naive Bayes. *IEEE* **2019**, *2019*, 1–6.
17. Merlo, A.; Migliardi, M.; Caviglione, L. A survey on energy-aware security mechanisms. *Pervasive Mob. Comput.* **2015**, *24*, 77–90. [CrossRef]
18. Fatima, A.; Maurya, R.; Dutta, M.K.; Burget, R.; Masek, J. Android Malware Detection Using Genetic Algorithm based Optimized Feature Selection and Machine Learning. *IEEE* **2019**, *2019*, 220–223.
19. Abawajy, J.H.; Kelarev, A. Iterative Classifier Fusion System for the Detection of Android Malware. *IEEE Trans. Big Data* **2019**, *5*, 282–292. [CrossRef]
20. Rais, H.M.; Mehmood, T. Dynamic Ant Colony System with Three Level Update Feature Selection for Intrusion Detection. *Int. J. Netw. Secur.* **2018**, *20*, 184–192.
21. Ren, Z.; Wu, H.; Ning, Q.; Hussain, I.; Chen, B. End-to-end malware detection for android IoT devices using deep learning. *Ad Hoc Netw.* **2020**, *101*, 102098. [CrossRef]
22. Cesare, S.; Xiang, Y.; Zhou, W. Control Flow-Based Malware VariantDetection. *IEEE Trans. Dependable Secur. Comput.* **2014**, *11*, 307–317. [CrossRef]
23. Bai, L.; Pang, J.; Zhang, Y.; Fu, W.; Zhu, J. Detecting Malicious Behavior Using Critical API-Calling Graph Matching. In Proceedings of the 2009 First International Conference on Information Science and Engineering, Nanjing, China, 26–28 December 2009.
24. Santos, I.; Devesa, J.; Brezo, F.; Nieves, J.; Bringas, P.G. OPEM: A Static-Dynamic Approach for Machine-Learning-Based Malware Detection. In Proceedings of the International Joint Conference CISIS'12-ICEUTE' 12-SOCO' 12 Special Sessions, Ostrava, Czech Republic, 5–8 September 2012.
25. Hughes, K.; Qu, Y. A Theoretical Model: Using Logistic Regression for Malware Signature Based Detection. 2012. Available online: https://www.researchgate.net/publication/271020436_A_Theoretical_Model_Using_Logistic_Regression_for_Malware_Signature_based_Detection (accessed on 6 April 2021).
26. Islam, R.; Tian, R.H.; Batten, L.M.; Versteeg, S. Classification of malware based on integrated static and dynamic features. *J. Netw. Comput. Appl.* **2013**, *36*, 646–656. [CrossRef]
27. Kilgallon, S.; De La Rosa, L.; Cavazos, J. Improving the Effectiveness and Efficiency of Dynamic Malware Analysis with Machine Learning. *IEEE* **2017**, *2017*, 30–36.
28. Kozachok, A.V.; Bochkov, M.V.; Kochetkov, E.V. Heuristic Malware Detection Mechanism Based on Executable Files Static Analysis. In Proceedings of the 3rd International Conference Information Technology and Nanotechnology 2017, Samara, Russia, 24–27 April 2017.
29. Morales-Molina, C.D.; Santamaria-Guerrero, D.; Sanchez-Perez, G.; Perez-Meana, H.; Hernandez-Suarez, A. Methodology for Malware Classification Using a Random Forest Classifier. In Proceedings of the 2018 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC), Ixtapa, Mexico, 14–16 November 2018.
30. Mohanasruthi, V.; Chakraborty, A.; Thanudas, B.; Sreelal, S.; Manoj, B.S. An Efficient Malware Detection Technique Using Complex Network-Based Approach. In Proceedings of the 2020 National Conference on Communications (NCC), Kharagpur, India, 21–23 February 2020.
31. Farrokhmanesh, M.; Hamzeh, A. Music classification as a new approach for malware detection. *J. Comput. Virol. Hacking Tech.* **2018**, *15*, 77–96. [CrossRef]
32. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In Proceedings of the 26th Annual Conference on Neural Information Processing Systems 2012, NIPS 2012, Lake Tahoe, NV, USA, 3–6 December 2012.
33. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B.S. *Malware Images: Visualization and Automatic Classification*; Association for Computing Machinery: New York, NY, USA, 2011; pp. 1–7.
34. Sun, G.; Qian, Q. Deep Learning and Visualization for Identifying Malware Families. *IEEE Trans. Dependable Secur. Comput.* **2021**, *18*, 283–295. [CrossRef]
35. Yue, S. *Imbalanced Malware Images Classification: A CNN Based Approach*; University of Wisconsin: Madison, WI, USA, 2017.
36. Rezende, E.; Ruppert, G.; Carvalho, T.; Ramos, F.; de Geus, P. Malicious Software Classification Using Transfer Learning of ResNet-50 Deep Neural Network. In Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017.
37. Li, W.; Zhang, R.; Wen, Q. A Malicious Code Variants Detection Method Based on Self-attention. In Proceedings of the 2020 6th International Conference on Computer and Technology Applications, Antalya, Turkey, 14–16 April 2020.

38. Priyadarsini, K.; Mishra, N.; Prasad, M.; Gupta, V.; Khasim, S. Detection of malware on the internet of things and its applications depends on long short-term memory network. *J. Ambient Intell. Humaniz. Comput.* **2021**, *1*, 1–12.
39. Nisa, M.; Shah, J.H.; Kanwal, S.; Raza, M.; Khan, M.A.; Damaševičius, R.; Blažauskas, T. Hybrid Malware Classification Method Using Segmentation-Based Fractal Texture Analysis and Deep Convolution Neural Network Features. *Appl. Sci.* **2020**, *10*, 4966. [CrossRef]
40. Damaševičius, R.; Venčkauskas, A.; Toldinas, J.; Grigaliūnas, Š. Ensemble-Based Classification Using Neural Networks and Machine Learning Models for Windows PE Malware Detection. *Electronics* **2021**, *10*, 485. [CrossRef]
41. Sezer, A.; Sezer, H.B. Capsule network-based classification of rotator cuff pathologies from MRI. *Comput. Electr. Eng.* **2019**, *80*, 106480. [CrossRef]
42. Hinton, G.; Sabour, S.; Frosst, N. Matrix capsules with em routing. In Proceedings of the ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018.
43. Yang, M.; Zhao, W.; Chen, L.; Qu, Q.; Zhao, Z.; Shen, Y. Investigating the transferring capability of capsule networks for text classification. *Neural. Netw.* **2019**, *118*, 247–261. [CrossRef] [PubMed]
44. Qiao, K.; Zhang, C.; Wang, L.; Chen, J.; Zeng, L.; Tong, L.; Yan, B. Accurate Reconstruction of Image Stimuli from Human Functional Magnetic Resonance Imaging Based on the Decoding Model With Capsule Network Architecture. *Front. Neurosci.* **2018**, *12*, 62. [CrossRef]
45. Afshar, P.; Plataniotis, K.N.; Mohammadi, A. Capsule Networks' Interpretability for Brain Tumor Classification Via Radiomics Analyses. In Proceedings of the 2019 IEEE International Conference on Image Processing, Taipei, Taiwan, 22–25 September 2019; pp. 3816–3820.
46. Jaiswal, A.; AbdAlmageed, W.; Wu, Y.; Natarajan, P. CapsuleGAN: Generative Adversarial Capsule Network. In *Computer Vision-Eccv 2018 Workshops, Pt Iii'*; Leal-Taixé, L., Roth, S., Eds.; Springer International Publishing Ag: Marina del Rey, CA, USA, 2019; pp. 526–535.
47. Andersen, P.A. *Deep Reinforcement Learning Using Capsules in Advanced Game Environments*; University of Agder: Grimstad, Norway, 2018.
48. Çayır, A.; Ünal, U.; Dağ, H. Random CapsNet forest model for imbalanced malware type classification task. *Comput. Secur.* **2021**, *102*, 102133. [CrossRef]
49. Wang, S.W.; Zhou, G.; Lu, J.C.; Zhang, F.J. A Novel Malware Detection and Classification Method Based on Capsule Network (China State Key Laboratory of Mathematical Engineering and Advanced Computer Zhengzhou 450001 China). *Artif. Intell. Secur.* **2019**, 573–584.
50. Phaye, S.; Sikka, A.; Dhall, A.; Bathula, D. *Dense and Diverse Capsule Networks: Making the Capsules Learn Better*; Indian Institute of Technology: Ropar, Punjab, India, 2018.
51. Ronen, R.; Radu, M.; Feuerstein, C.; Yom-Tov, E. *Microsoft Malware Classification Challenge*; Researchgate: Berlin, Germany, 2018.
52. Zhao, Y.T.; Cui, W.J.; Geng, S.N.; Bo, B.; Feng, Y.X.; Zhang, W.B. A Malware Detection Method of Code Texture Visualization Based on an Improved Faster RCNN Combining Transfer Learning. *IEEE Access* **2020**, *8*, 166630–166641. [CrossRef]
53. Foster, D.J. *Artificial Intelligence in Cybersecurity: A Novel Deep Learning Architecture for Image-Based*; University of London: London, UK, 2020.
54. Guo, X. A Keras Implementation of CapsNet. Available online: https://github.com/XifengGuo/CapsNet-Keras/tree/tf2.2 (accessed on 6 April 2021).
55. Ahmadi, M.; Ulyanov, D.; Semenov, S.; Trofimov, M.; Giacinto, G. Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification. In Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, New York, NY, USA, 9–11 March 2016.
56. Narayanan, B.N.; Djaneye-Boundjou, O.; Kebede, T.M. *Performance Analysis of Machine Learning and Pattern Recognition Algorithms for Malware Classification*; IEEE: New York, NY, USA, 2016; pp. 338–342.
57. Fu, J.; Xue, J.; Wang, Y.; Liu, Z.; Shan, C. Malware Visualization for Fine-Grained Classification. *IEEE Access* **2018**, *6*, 14510–14523. [CrossRef]
58. Cui, Z.; Xue, F.; Cai, X.; Cao, Y.; Wang, G.G.; Chen, J. Detection of Malicious Code Variants Based on Deep Learning. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3187–3196. [CrossRef]
59. Gibert, D.; Mateu, C.; Planes, J.; Vicens, R. Using convolutional neural networks for classification of malware represented as images. *J. Comput. Virol. Hacking Tech.* **2019**, *15*, 15–28. [CrossRef]
60. Narayanan, B.N.; Davuluru, V.S.P. Ensemble Malware Classification System Using Deep Neural Networks. *Electronics* **2020**, *9*, 721. [CrossRef]
61. Roseline, S.A.; Geetha, S.; Kadry, S.; Nam, Y. Intelligent Vision-Based Malware Detection and Classification Using Deep Random Forest Paradigm. *IEEE Access* **2020**, *8*, 206303–206324. [CrossRef]