mathematics

MDPI

Article

Malware Detection Based on the Feature Selection of a Correlation Information Decision Matrix

Kai Lu [1,2], Jieren Cheng [3,*] and Anli Yan [1]

[1] School of Cyberspace Security (School of Cryptology), Hainan University, Haikou 570100, China
[2] Department of Public Safety Technology, Hainan Vocational College of Political Science and Law, Haikou 571100, China
[3] School of Computer Science and Technology, Hainan University, Haikou 570100, China
* Correspondence: hnplctx@163.com

Abstract: Smartphone apps are closely integrated with our daily lives, and mobile malware has brought about serious security issues. However, the features used in existing traffic-based malware detection techniques have a large amount of redundancy and useless information, wasting the computational resources of training detection models. To overcome this drawback, we propose a feature selection method; the core of the method involves choosing selected features based on high irrelevance, thereby removing redundant features. Furthermore, artificial intelligence has implemented malware detection and achieved outstanding detection ability. However, almost all malware detection models in deep learning include pooling operations, which lead to the loss of some local information and affect the robustness of the model. We also propose designing a malware detection model for malicious traffic identification based on a capsule network. The main difference between the capsule network and the neural network is that the neuron outputs a scalar, while the capsule outputs a vector. It is more conducive to saving local information. To verify the effectiveness of our method, we verify it from three aspects. First, we use four popular machine learning algorithms to prove the effectiveness of the proposed feature selection method. Second, we compare the capsule network with the convolutional neural network to prove the superiority of the capsule network. Finally, we compare our proposed method with another state-of-the-art malware detection technique; our accuracy and recall increased by 9.71% and 20.18%, respectively.

Keywords: feature selection; capsule network; malware detection; network traffic

MSC: 68T05

Citation: Lu, K.; Cheng, J.; Yan, A. Malware Detection Based on the Feature Selection of a Correlation Information Decision Matrix. *Mathematics* **2023**, *11*, 961. https://doi.org/10.3390/math11040961

Academic Editors: Zibin Zheng, Ruoxi Jia, Dan Li, Yuxun Zhou and Liang Xu

Received: 9 January 2023
Revised: 3 February 2023
Accepted: 10 February 2023
Published: 13 February 2023

1. Introduction

With the popularity of the internet and mobile devices, malware has become a major threat to the growing mobile ecosystem. Kaspersky's statistical report [1] shows that in 2020, the number of new malicious files detected every day reached 360,000, an increase of 5.2% compared with the previous year. Although mobile anti-virus scanners provide security protection mechanisms for Android devices, more advanced mobile malware can still infiltrate mobile systems by circumventing these mechanisms. Given that mobile devices are carrying an increasing amount of users' private information, there is an urgent need to develop an efficient malware detection scheme.

Malware detection technology can be divided into three types: static analysis, dynamic analysis, and network traffic analysis. The essential difference between these three types of methods is that they use different features [2]. Static analysis methods use application codes and binary structures as features [3,4]. However, to avoid detection by anti-virus scanners, malware authors use techniques, such as repackaging and code obfuscation to generate malware variants. The 'calling relationship' between functions characterizes the dynamic analysis method during the running of the application [5]. This method needs to

be completed on a specific sandbox and requires sufficient execution to cover the behavior of the application. When the malware author repackages the malware or obfuscates the code, the features of the methods as mentioned earlier change dramatically, leading to a decrease in the performance of the detection model. From another point of view, these malware variants have similar malicious behaviors. In other words, malicious traffic triggered by malware is similar. Network traffic analysis takes the network traffic triggered by the application as the research object [6]. This method extracts statistical features (such as packet size and packet interval) [7] or HTTP header semantic features (such as host and method) [8] from the network traffic for analysis. Therefore, the network traffic analysis method overcomes the shortcomings of static and dynamic analyses, because even if the malicious code changes significantly, some traffic features are similar. This paper uses the research object by Wang et al.'s [9] public network traffic dataset.

Machine learning provides multiple methods to deal with malware detection. If the only goal is to accurately detect malware, deep learning is usually a better choice [6]. Research has shown that, compared with other machine learning technologies, deep learning has demonstrated excellent performance in different application fields [10]. Similarly, deep learning has also been researched in the field of malware detection and has achieved high performance [11]. However, the algorithms used in deep learning for malware detection models are almost all based on convolutional neural networks. Through pooling operations, convolutional neural networks are helpful for analysis, but some local information is lost, resulting in a decrease in robustness. To overcome the shortcomings of convolutional neural networks, we conducted verification experiments on the capsule network as a malware detection model. A capsule network is more conducive to preserving local information and having local equivariant characteristics, so it can better improve its generalization ability. In addition, we also propose a feature selection method to further filter the features of the dataset, thereby further improving the detection performance of the model. The contributions made in this paper are summarized as follows:

- We propose a novel malware detection method integrating feature selection and a malware detection model. The malware detection model is a capsule network that overcomes the deficiencies of a convolutional neural network. To the best of our knowledge, this is the first time that a capsule network has been applied in the field of malware detection.
- We propose a feature selection method based on a correlation information decision matrix to reduce the dimensions of high-dimensional data and provide a strong guarantee for subsequent detection tasks.
- We evaluated the effectiveness of our method through some detailed experiments and compared our method with state-of-the-art malware detection techniques. Our experiments show that our method achieves higher detection results.

The rest of the paper is organized as follows: Section 2 introduces related works. Section 3 presents our proposed method. Section 4 describes the experimental design and implementation. Section 5 discusses the experimental results. Section 6 concludes the paper.

2. Related Work

2.1. Feature Selection

The form of feature selection can be divided into two categories: filter and model-based. The filter involves scoring each feature according to divergence or correlation and then setting a threshold or the number of features to be selected for filtering. Jemal et al. [12] defined feature selection as a quadratic programming problem, and analyzed how the filter-based feature selection method works in Android malware detection. Mouhammd et al. [13] studied feature selection and malware classification based on machine learning. These features are identified intuitively. The features of each part of the PE (portable executable) file can be associated with each other, not with class files. Pushkar et al. [14] proposed a

new JavaScript analysis and detection technology based on the sandbox assisted ensemble model, and calculated the Pearson coefficient between each feature to extract features.

The working mode of the model-based feature selection algorithm can be divided into two cases. One is to train the model on the training set for each feature subset to be selected and then select the feature subset on the test set according to the size of the error. The other first uses some machine learning models for training, obtains the weight coefficients of each feature, and finally selects the features according to the coefficients from large to small. Yan et al. [15] proposed a two-tier architecture for malware detection, which uses a random forest algorithm for feature selection before training the detection model. Csahin et al. [16] used a feature selection method based on linear regression to remove unnecessary features, and then built a malware detection system. Anam et al. [17] proposed a machine learning-based Android malware detection method, using an evolutionary genetic algorithm for feature selection. The results show that the feature selection still keeps the classification accuracy of over 94%, and reduces the feature dimension greatly. Harry et al. [18] proposed an improved feature learning method for malware detection, which is based on Aminanto et al. [19]. The authors of [19] proposed deep abstraction and weighted feature selection (DFES) for intrusion detection systems in 2017.

Compared with the above feature selection work, we propose a feature selection method based on the correlation information decision matrix (CIDM) to reduce the dimension of high-dimensional data. CIDM can keep the original dimension information. In addition, based on the feature selection method, CIDM achieves soft dimensionality reduction through iterative statistical analysis of the correlation coefficient matrix.

2.2. Malware Detection-Based Network Traffic

As this paper focuses on network traffic, we only analyze the detection of malware based on network traffic. The different features extracted from network traffic can be divided into textual feature and statistical feature analyses [15].

Textual feature analysis involves extracting text information in traffic, such as the URL, packet header, etc., and then using natural language processing technology to preprocess it and convert it into a data format that can be processed by a machine learning model [20]. Li et al. [21] proposed a method combining linear space transformation and nonlinear space transformation. For linear space transformation, it first performs singular value decomposition to obtain the orthogonal space and then uses linear programming to solve the optimal distance metric. For nonlinear space transformation, they introduced the Nyström method [22] for kernel approximation and adopted a modified distance metric for its radial basis function. Wang et al. [20] proposed a method to identify malware by using a URL accessed by an application. In addition, they established a multi-view neural network, which can automatically generate multiple input views and assign soft attention weights to different input features. Taiga et al. [23] proposed a malware detection method that does not require benign communication traffic. This method generates a template containing the HTTP request templates generated by the malware. Then, it detects whether the host is infected by malware by comparing the monitored HTTP request set with the template set. Kitsune [24] is a plug-and-play unsupervised network intrusion detection system. Kitsune's core algorithm uses an integrated autoencoder to distinguish between benign and malicious traffic patterns.

Statistical feature analysis involves extracting statistical information from the traffic, such as data packet size, the time interval between data packets, etc., and then normalizing the extracted statistical features to train the machine learning model [25]. Wang et al. [26] proposed a lightweight security mechanism to detect malicious traffic, which is based on the Chebyshev polynomial approximation theory of time series. Cheng et al. [27] proposed a deep packet inspection (OFDPI) method based on the SDN paradigm to provide adaptive and efficient packet inspection. First, OFDPI checks the IP address of each new flow through the OpenFlow protocol, which provides for early detection at the flow-level granularity. Then, OFDPI allows for deep packet inspection at the packet-level granularity. A malicious

traffic detection framework called Malfinder—based on ensemble learning—was proposed in [28]. Malfinder uses statistical features and sequence features to describe network traffic and extends the dimensions of these two features to enhance their ability to represent traffic data. When considering that most of the network traffic is benign and only a small part is malicious, which leads to the data imbalance issue. Chen et al. [29] proposed solving the challenge of low model prediction accuracy caused by data imbalance issues. They use the imbalanced data gravitation-based classification (IDGC) algorithm to classify imbalanced data.

Compared with the above malware detection-based network traffic work, we propose a malware detection method based on a capsule neural network. It is also the first application of a capsule neural network in the field of malware detection since it was proposed in 2017 [30]. In addition, to make the capsule neural network achieve better detection performance, we also developed a feature selection algorithm based on feature correlation.

3. Methodology

Our proposed malware detection method includes two main parts, one is feature selection and the other is the detection model. It operates by (1) extracting more vital features to reduce feature dimensions based on the raw text features of network traffic, and (2) detecting malicious traffic through the capsule network.

3.1. Feature Selection

For high-dimensional traffic data, the necessary dimension reduction preprocessing can improve the efficiency of subsequent processing and the accuracy of recognition. Before classification work, to retain the direct information of the original data and refine the dimension reduction process, we propose a feature selection method based on a correlation–information decision matrix (CIDM) to reduce the dimension. The factors we consider are the correlation between features and the amount of information on features. Generally speaking, the lower the correlation between features, the more information they carry. Before introducing the feature selection method, several related contents (such as CIDM and the score of features) need to be introduced.

3.1.1. Correlation Decision Matrix

The reference information that determines which features are redundant is obtained from the CIDM; the CIDM is optimized from the correlation decision matrix (CDM). The generation process of CDM is shown in Figure 1. First, we use Equation (1) to calculate the correlation coefficient between each pair of features and obtain the correlation coefficient matrix C, where $Var(A_i)$ is the variance between the values within the feature A_i, which is computed by Equation (2), M is the number of features and μ_i is the elements average value of A_i. Additionally, $Cov(A_iA_j)$, as is shown in Equation (3), is the covariance between features A_i and A_j. In these Equations, $1 \leq i, j \leq M$. If the value of the elements in matrix C is less than 0, it is converted to its opposite number, i.e., the elements in the matrix are all non-negative. Second, an initial correlation decision matrix O for matrix C is established. Currently, each element value of each row of matrix O corresponds to the serial number of its column. For example, the value of the $i-th$ column from the left is i. Next, each row element of the matrix O is arranged in ascending order according to each row's value of the corresponding matrix C to obtain CDM O'. Finally, we conduct the statistical analysis of this iteration in the local matrix (red box range) with *width* to determine which features are reduced.

$$\rho_{A_iA_j} = \frac{Cov(A_iA_j)}{\sqrt{Var(A_i)Var(A_j)}} \tag{1}$$

$$Var(A_i) = \frac{1}{n}\sum_{i=1}^{M}(A_i - \mu_i)^2 \tag{2}$$

$$Cov(A_iA_j) = \frac{1}{n}\sum_{k=1}^{M}(A_{ik}A_{jk}) - \frac{1}{n^2}(\sum_{1}^{M}A_{ik})(\sum_{1}^{M}A_{jk}) \tag{3}$$

In addition, if we only consider the correlation for dimensionality reduction, we will reduce the main features in some extreme cases, for example, there are three vectors: a = [1,0,0,0,0,0], b = [0,1,0,0,0,0], c = [1,1,1,0,1,0]. $corrcoef(x, y)$ is the correlation coefficient function of the two vectors x and y. Then, corrcoef(a,b) = 0.2, corrcoef(a,c) = corrcoef(b,c) = 0.32. According to the rule that the higher the correlation between features is, the lower the amount of information they carry, the features c with high amounts of data should be reduced. This is obviously wrong. It is similar to two low information features, a and b, which crowd out their differences c. To avoid this situation, in fact, it does happen in the data used in the latter experiment, the consideration of the amount of information is essential. The information $aoi(i)$ is represented by the number of non-zero elements of feature i. All elements in the matrix C are adjusted by Equations (4) and (5).

$$C_{ij} = \frac{C_{ij}}{\gamma} + \frac{(\gamma - 1) * aoi\_r(j) * ave\_C}{\gamma * ave\_aoi\_r}, 1 \le i, j \le M \tag{4}$$

$$aoi\_r(i) = \begin{cases} N/aoi(i), & aoi(i)/N < a \\ N/(N - aoi(i)), & aoi(i)/N \ge a \end{cases} \tag{5}$$

where $\frac{ave\_C}{ave\_aoi\_r}$ is a weighting factor, which is used to adjust the correlation and information range to the same order of magnitude in Equation (4). ave_C is the average of all elements of the matrix C and ave_aoi_r represents the average of aoi_r. γ is used to control the proportion of the correlation and information in the assignment and the value range is $[1, +\infty)$. From Equation (4), it is obvious that when γ is equal to 1, only the correlation is considered; when it is oriented to positive infinity, only the amount of information is considered. Parameter a is used to control the intensity of low information feature selection; here, we use 0.9. Now, we can obtain the matrix CIDM according to the flow of Figure 1.

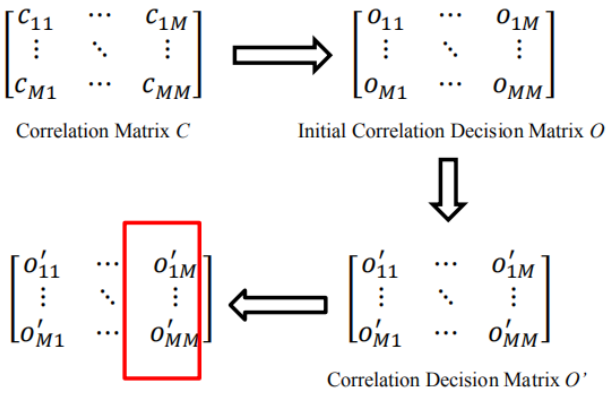

Figure 1. Establishment of CDM.

3.1.2. Scoring of Attributes

The local matrix consists of the *width* column and M rows. In this matrix, the frequency of each existing feature is counted, and the average and variance of the correlation coefficient with all features are combined to score. This score value is used as the basis for judging feature reduction. We sort these scores, and the first *width* features with the largest scores are taken as the reduction objects in this iteration. Instead of using the frequency of occurrence as the judgment basis, the mean value and variance of the correlation coefficients between features are added to avoid ambiguity in the selection of reduction objects caused by the same frequency of occurrence among multiple features. Specifically, the score equation is shown in Equation (6), where $Ave(C_i)$ and $Var(C_i)$ represent the average and variance of the i-th row of matrix C, and $S\_score(A_i)$ is the statistical frequency, which is the score of attribute A_i in the local matrix in the current iteration.

$$Score(A_i) = Ave(C_i) + Var(C_i) + S\_score(A_i) \tag{6}$$

3.1.3. Algorithm of Dimension Reduction

According to CDM and the feature scoring method, a feature selection method based on CDM is proposed. The specific execution process pseudo-code of the dimension reduction algorithm is shown in Algorithms 1 and 2.

Algorithm 1 Iterative feature selection.

Require: Dataset $X = [x_1, x_2, \ldots, x_M]$, $width$, $goal\_dim$.
Ensure: $X' = [x'_1, x'_2, \ldots, x'_{goal\_dim}]$

Establish the correlation matrix C by Equations (1)–(5)
The elements that are less than 0 in matrix C are replaced by their absolute values
$C = C - E$
$iteration \leftarrow floor((M - goal\_dim)/width)$
$remainder \leftarrow (M - goal\_dim) - width * iteration$
while $iteration > 0$
    $X \leftarrow Score(X, width)$
    $iteration \leftarrow iteration - 1$
end while
if $remainder \neq 0$
    $width \leftarrow remainder$
    $X \leftarrow Score(X, width)$
end if
$X' \leftarrow X$

Algorithm 2 Score.

Require: $X = [x_1, x_2, \ldots, x_M]$, $width$.
Ensure: X after dimension reduction in this iteration

Establish the correlation matrix C by Equations (1)–(3)
The elements less than 0 in matrix C are replaced by their absolute values
$C \leftarrow C - E$
Calculate $Ave(C_i)$ and $Var(C_i)$
Establish correlation decision matrix O'
Count $S\_score(A_i)$ in the local matrix with width $width$
Calculate $Score(A_i)$ by Equation (6)
Sort the attribute number by the corresponding $Score$ value
Reduce the $width$ dimensions of X with the largest $Score$ value

Through the description of the algorithm pseudo-code, the whole dimension reduction process can be cleared. In the details of Algorithm 1, the parameter $goal\_dim$ represents the number of dimensions to reduce dataset X. The purpose of setting this parameter is to match the input interface of the subsequent classification model. E is an identity matrix of M rows and M columns; $iteration$ determines the number of iterations in the whole reduction process; $remainder$ is the remainder of the differences between the initial data dimensions M and $goal\_dim$ divided by $width$. The parameter $width$ is determined by the experiments. If $remainder \neq 0$ means that the algorithm completes the $iteration$ reduction processes, there are $remainder$ (smaller than $iteration$) attributes that need to be reduced. The reason why the local matrix is used to score features in each iteration is to make the trade-off between multiple features with high correlations clear. For example, in extreme cases, there is a high correlation between two features. At this time, only one of them needs to be reduced. How does one choose between them? The statistical score based on the local matrix can solve this problem well.

3.2. Malware Detection

Our malware detection model is based on a capsule network. A capsule neural network was first proposed by Sabour et al. [31]. The main difference between a capsule network and a neural network is that the neuron outputs a scalar, while the capsule outputs a vector. The simple structure diagram is shown in Figure 2.

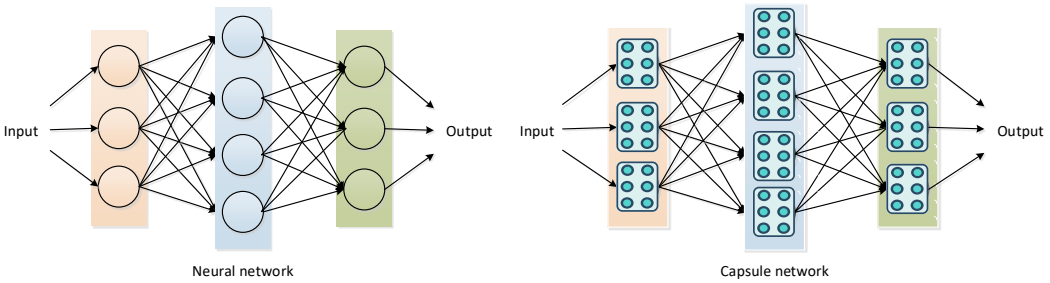

Figure 2. A simple comparison between the capsule network and neural network in the structure.

The reason why we adopt a capsule network is that it overcomes the deficiencies in a convolutional neural network. A convolutional neural network obtains invariance through the pooling operation, which is helpful for the analysis. However, some local information is lost. The solution to this problem is data enhancement, which generates a new training set by rotating and shifting the training samples. Unfortunately, for the network traffic dataset, it is meaningless to rotate and shift it. In other words, the network traffic after the rotation and shift cannot correctly describe the flow. Therefore, we use a capsule network to overcome the shortcomings of the convolutional neural network, to obtain a more robust detection model.

To introduce the capsule network more clearly, we describe it from three aspects: the operation mechanism of a single capsule, the core algorithm of the capsule network (dynamic routing), and the loss function.

3.2.1. The Operation of a Capsule

The operation process of a capsule is shown in Figure 3, in which the output of three capsules is taken as the input of the next capsule. The output vectors of the three capsules, v^1, v^2, and v^3, are used as the input of the next capsule; v^1, v^2, and v^3 are multiplied by the other matrices (w^1, w^2, and w^3) to obtain u^1, u^2, and u^3, respectively. Next, u^1, u^2, and u^3 are the weighted sums used to obtain s Then, v is obtained by squashing. Squash only changes the length, it does not change the direction. Parameters w^1 and w^2 are obtained through backpropagation learning. Moreover, c^1, c^2, and c^3 are called coupling coefficients. They use the dynamic decisions of capsules when testing. This decision-making process is called dynamic routing, and the details are described in Section 3.2.2.

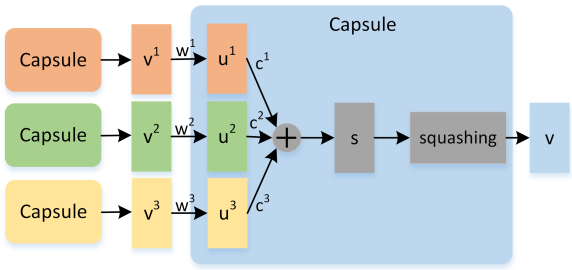

Figure 3. The operation process of a capsule.

The calculation equation of u is shown in Equation (7). i in the equation is the index of capsules. For a simple example of Figure 3, $u^1 = v^1 w^1$, $u^2 = v^2 w^2$, and $u^3 = v^3 w^3$.

$$u^i = v^i w^i \tag{7}$$

s is obtained by summing the weights of u^i, and the calculation equation of s is shown in Equation (8), n is the number of capsules in the upper layer. For a simple example of Figure 3, n is 3. According to Equation (8), the expansion of s is calculated as $s = u^1c^1 + u^2c^2 + u^3c^3$.

$$s_i = \sum_{i=1}^{n} u^i c^i \tag{8}$$

The calculation of capsule output v by s needs an operation called squash. The specific calculation of squash is shown in Equation (9). In Equation (9), $||\cdot||^2$ is the norm of the vector.

$$v = Squash(s) = \frac{||s||^2}{1 + ||s||^2} \frac{s}{||s||^2} \tag{9}$$

3.2.2. Dynamic Routing

The c^1, c^2, and c^3 choices are determined by the dynamic routing algorithm. The pseudo-code of the dynamic routing algorithm is shown in Algorithm 3. First of all, there must be the parameter B set; the initial values of B are all zeros, and $\{b_0^1, b_0^1, b_0^1, \ldots, b_0^i\}$ correspond to $\{c^1, c^2, c^3, \ldots, c^i\}$. Suppose you run T iterations, and T is a predetermined hyperparameter.

Algorithm 3 Dynamic routing.

Require: B = $\{b_0^1, b_0^2, b_0^3, \ldots, b_0^i\}$, U = $\{u^1, u^2, u^3, \ldots, u^i\}$← B is a set of parameters, U is the output of a layer of the capsule network.
Ensure: $C_T = \{c_T^1, c_T^1, c_T^1, \ldots, c_T^i\}$
1: for r = 1 to T do
2: $c_r^1, c_r^1, c_r^1, \ldots, c_r^i$ = softmax$(b_{r-1}^1, b_{r-1}^1, b_{r-1}^1, \ldots, b_{r-1}^i)$
3: $s_r = \sum_{i=1}^{n} u^i c_r^i$
4: $a_r = Squash(s_r)$
5: $b_r^i = b_{r-1}^i + a_r u^i$
6: end for

For a more convenient and intuitive understanding, we give an example, as shown in Figure 4. We have v^1, v^2 and v^3, and we multiply by w^1, w^2 and w^3 to obtain $U = \{u^1, u^2, u^3\}$. We carry out the routing algorithm, refer to Algorithm 3. First, we initialize $B_0 = \{b_0^1, b_0^2, b_0^3\}$, they are all zeros, T is 3. Next, according to B_0, we can obtain $C_1 = \{c_1^1, c_1^2, c_1^3\}$, as shown on line 2. With C_1, we can calculate a s_1 and obtain a a_1 as shown on lines 3–4. According to a_1, we can decide the next round of $C_2 = \{c_2^1, c_2^2, c_2^3\}$. Afterward, through line 5, we update parameter B, $B_1 = \{b_1^1, b_1^2, b_1^3\}$, where $b_1^1 = b_0^1 + a_1u^1$, $b_1^2 = b_0^2 + a_1u^2$ and $b_1^3 = b_0^3 + a_1u^3$. After the first iteration, the second iteration is performed according to B_1. We use B_1 to obtain C_2. We can calculate s_2 by C_2, and then we can obtain a_2. Subsequently, we update parameter B, $B_2 = \{b_2^1, b_2^2, b_2^3\}$, where $b_2^1 = b_1^1 + a_2u^1$, $b_2^2 = b_1^2 + a_2u^2$ and $b_2^3 = b_1^3 + a_2u^3$. After the second iteration, the last iteration is performed according to B_2. The process is calculated from B_2 to C_3, from C_3 to s_3, and from s_3 to a_3 = v.

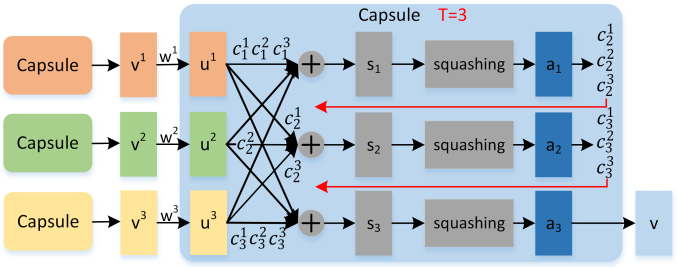

Figure 4. The operation process of a capsule.

3.2.3. Loss Function

Capsule neural network provides two loss functions, one is margin loss, which is used for the classification task, and the other is reconstruction loss, which is used for sample reconstruction. Since our task is to detect malicious traffic, margin loss is adopted. The equation of the margin loss function is shown in Equation (10).

$$L_k = E_k max(0, m^+ - ||v_k||)^2 + \lambda(1 - E_k)max(0, ||v_k|| - m^-)^2 \quad (10)$$

where E_k is the existence of k class, the existence is 1, and the nonexistence is 0. m^+ is 0.9, the penalty false is positive, the k class exists but prediction does not exist, m^- is 0.1, the penalty false is negative, the k class does not exist but prediction exists. λ is the weight.

The specific architecture information of the capsule neural network and CNN is described in https://download.csdn.net/download/littlle_yan/87399101 (accessed on 8 January 2023).

4. Experimental Design and Implementation

4.1. Datasets

We use the URL extracted from the network traffic published by Wang et al. [9]. For the network traffic collection method, Wang et al. used the Android tool monkey to send some events randomly to the device to trigger network traffic during the execution of each application. To avoid network traffic being mixed with different applications, they executed only one application at a time. This dataset provides information on the method, host, page, and name fields in the URL. Each sample is represented by 1708 features. The specific numbers of benign traffic and malicious traffic are shown in Table 1. Our work feature selections are based on 1708 features per sample.

Table 1. The specific information about the dataset.

Label	NO.
Benign	25,276
Malicious	11,251

4.2. Experiment Setup

We used the Python language. We mainly used the TensorFlow framework and sklearn library. The experiments were conducted on a server with an Intel Core i5-8500 CPU @ 3.00 GH and 8 GB RAM running Ubuntu 14.04. The optimizer of the training capsule network and CNN was Adadelta.

4.2.1. Parameter Setting Analysis

width and γ are crucial parameters in the dimension reduction of this paper. Different parameter values affect the efficiency of dimensionality reduction. To obtain the appropriate parameter values in the dataset environment, and ensure high-dimensional reduction efficiency and a stable-dimensional reduction process, corresponding experiments for setting *width* and γ were designed.

4.2.2. Feature Analysis

The purpose of feature analysis is to verify the effectiveness of our proposed dimensionality reduction method. To achieve this goal, we used classification algorithms on data without dimensionality reduction and data with dimensionality reduction. We used the four most popular algorithms, decision tree (DT), random forest (RF), logistic regression (LR), and K-nearest neighbor (KNN).

4.2.3. Model Analysis

The model analysis focuses on the evaluation of the capsule network, showing the performance differences between the capsule network and other deep learning networks. We chose the convolution neural network (CNN) as the contrast object. To ensure fairness and rationality, all methods used the same training set and test set in the evaluation experiment.

4.2.4. Comprehensive Analysis

To further verify the effectiveness of the method, we compared our method with other state-of-the-art malware detection techniques [9] in the comprehensive analysis. Wang et al. [9] proposed a detection method based on the floating centroid method (FCM), which combines supervised classification and unsupervised clustering.

4.3. Evaluation Metrics

The evaluation metrics we use are accuracy, precision, recall, and F-measure, which are calculated based on a confusion matrix. The confusion matrix is shown in Table 2. In the table, TP is truly positive, which means that the real label of the sample is positive, and the result predicted by the model is also positive. TN is a true negative, which means that the real label of the sample is negative, and the model predicts it to be negative. FP is the false positive, which means that the real label of the sample is negative, but the model predicts it to be positive. FN is the false negative, which means that the real label of the sample is positive, but the model predicts it as negative. The equations for the four metrics we used are shown below.

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

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

$$FPR = \frac{FP}{TN + FP} \tag{14}$$

$$F - measure = \frac{2 \times precision \times recall}{precision + recall} \tag{15}$$

Table 2. Confusion matrix.

Real Label	**Prediction Label**	
	Positive	**Negative**
positive	TP	FN
negative	FP	TN

5. Evaluation

5.1. Parameter Setting Analysis

The dataset dimension was 1708, and the input interface of the capsule network required 784-dimensional data, i.e., $goal\_dim = 784$; then, 924 attributes needed to be reduced by the algorithm. Firstly, the running times of the dimension reduction algorithm under different parameter (*width*) values were tested; the parameter *width* value sets are in $[5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70]$. Phenomena can be seen in Figure 5; with the increase in the *width* value, the time cost of the dimension reduction gradually decreases and tends to be stable. Based on this figure, the condition to select the parameter value is that the running time should be as low as possible and the value of *width* should be as small as possible. Next, Figure 6 shows the effect of the *width* on the dimensionality reduction. The value of the ordinate indicates the number of common features in the dimension reduction sequence corresponding to the adjacent values of the *width*. Different parameter values can lead to different dimensionality reduction sequences, such as in Figure 7, the values in the figure indicate the initial feature number. Figure 7 shows the partial results of the dimension reduction sequence corresponding to different parameters. It can be seen from Figure 6 that the values of the *width* parameter have no obvious influence on the overall dimension reduction results, but it is clearly from Figure 7 that the dimension reduction sequences corresponding to values of *width* are obviously different. The smaller the parameter value is, the more detailed the relationship between dimensions can be reflected. Finally, through the above experimental analysis, we decide that the value of the *width* is 40. The reason is that its value is very small relative to the total dimension number (924) to be reduced, and the dimension reduction efficiency is also high at this time. Therefore, the dimension reduction result is obtained when the *width* value equal to 40 is the input sample set of the subsequent capsule network model.

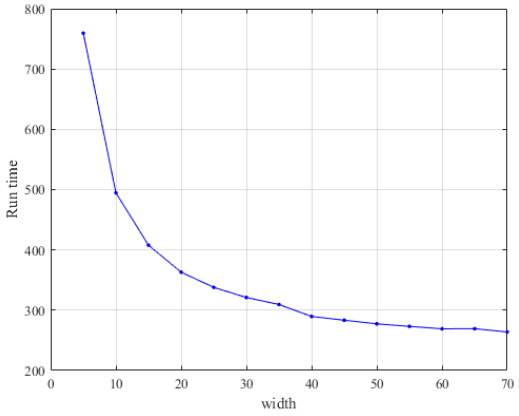

Figure 5. Run time for dimensionality reduction for various values of *width*.

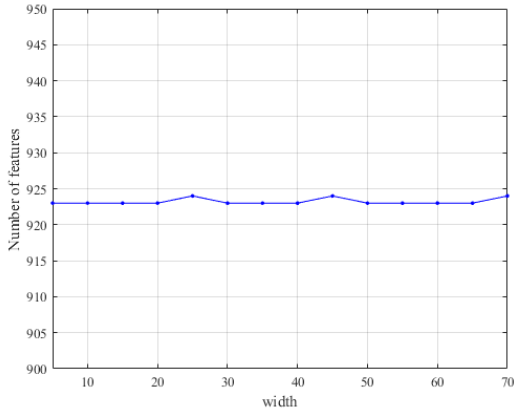

Figure 6. The same number of reduced features under adjacent *width* settings.

width=5	width=10	width=15	width=20	width=25	width=30	width=35	width=40	width=45	width=50	width=55	width=60	width=65	width=70
1684	1687	1702	1702	1706	1708	1708	1708	1708	1708	1708	1708	1708	1708
1666	1684	1687	1691	1702	1706	1706	1706	1706	1706	1706	1706	1706	1706
1197	1676	1684	1687	1694	1705	1705	1705	1705	1705	1705	1705	1705	1705
1115	1666	1676	1684	1691	1704	1704	1704	1704	1704	1704	1704	1704	1704
531	1197	1666	1681	1687	1702	1703	1703	1703	1703	1703	1703	1703	1703
1687	1115	1556	1676	1684	1699	1702	1702	1702	1702	1702	1702	1702	1702
1676	1108	1229	1666	1681	1694	1699	1700	1700	1700	1700	1700	1700	1701
1108	531	1197	1643	1676	1691	1698	1699	1699	1699	1699	1699	1699	1700
519	519	1115	1556	1669	1687	1695	1698	1698	1698	1698	1698	1698	1699
4	4	1108	1515	1666	1684	1694	1697	1697	1697	1697	1697	1697	1698
1702	1702	531	1229	1649	1681	1691	1695	1695	1696	1696	1696	1696	1697
1556	1691	519	1197	1643	1676	1690	1694	1694	1695	1695	1695	1695	1696
1229	1681	400	1115	1639	1669	1689	1692	1692	1694	1694	1694	1694	1695
400	1643	331	1108	1556	1667	1687	1691	1691	1693	1693	1693	1693	1694
331	1556	4	531	1515	1666	1684	1690	1690	1692	1692	1692	1692	1693
1691	1515	1708	519	1229	1649	1681	1689	1689	1691	1691	1691	1691	1692
1681	1229	1706	400	1197	1643	1676	1687	1687	1690	1690	1690	1690	1691
1643	400	1705	345	1115	1639	1669	1684	1684	1689	1689	1689	1689	1690
1515	345	1704	331	1108	1556	1667	1681	1681	1687	1687	1687	1687	1689
345	331	1699	4	531	1515	1666	1676	1680	1684	1684	1684	1684	1687
1706	1708	1694	1708	519	1229	1649	1669	1676	1681	1681	1681	1681	1684
1694	1706	1691	1706	400	1197	1643	1667	1675	1680	1680	1680	1680	1681
1669	1705	1681	1705	345	1115	1639	1666	1669	1676	1679	1679	1679	1680
1649	1704	1669	1704	331	1108	1556	1649	1667	1675	1676	1676	1676	1679
1639	1699	1667	1703	4	531	1515	1643	1666	1669	1675	1675	1675	1678
1708	1694	1649	1700	1708	519	1229	1639	1663	1667	1669	1669	1669	1676

Figure 7. The change of the dimension reduction sequence under a different *width*.

γ is another important parameter. Its value determines the proportion of the correlation and information in the dimension reduction. The value of γ is determined by the classification effect of the dimension reduction results in the decision tree. The classification effect is evaluated by the four indexes: accuracy, precision, recall, and F-measure. Under the condition that the *width* is 40 and each index reaches the highest, the lowest γ is the final value we want. The specific experimental results are shown in Figure 8.

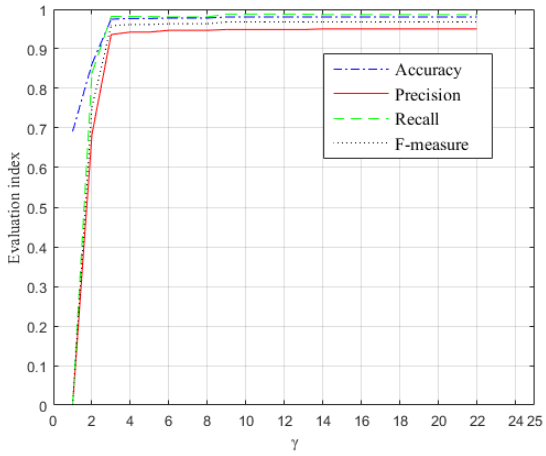

Figure 8. The classification indexes under various values of γ.

In the experiment, the value of γ starts from 1 and is tested every 0.5. When γ is equal to 7.5, each index reaches the best, so we take the dimension reduction result for γ = 7.5 as the data of the subsequent classification model.

5.2. Feature Analysis

To analyze the performance of our feature selection method, we used four popular machine learning algorithms to train on the datasets with feature selection; at the same time, we also used these four popular machine learning algorithms to train on the datasets without feature selection. The machine learning algorithms we used were decision tree (DT), random forest (RF), logistic regression (LR), and K-nearest neighbor (KNN). As shown in Figures 9–12, different algorithms have different performances. In the figure, the abscissa represents the different algorithms, and the ordinate represents the metric values. From the global analysis, it is not difficult to find that the detection performance of the models trained with the dimensionality reduction dataset is not lower than those models without dimensionality reduction, especially the RF algorithm, the precision with dimensionality reduction is 4.97% higher than that without dimensionality reduction, the KNN algorithm,

the recall with dimensionality reduction is 3.71% higher than that without dimensionality reduction. The reason for this phenomenon is that the data for dimensionality reduction does not cause the loss of data information due to the decline of feature dimension but removes the counterproductive features. Through experiments and results analyses, we can conclude that our proposed dimensionality reduction method plays a positive role, so our method is effective.

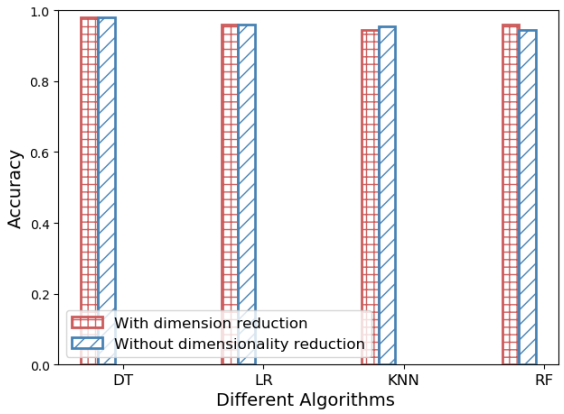

Figure 9. Accuracy comparison between different algorithms in the dimensionality reduction and non-dimensionality reduction datasets.

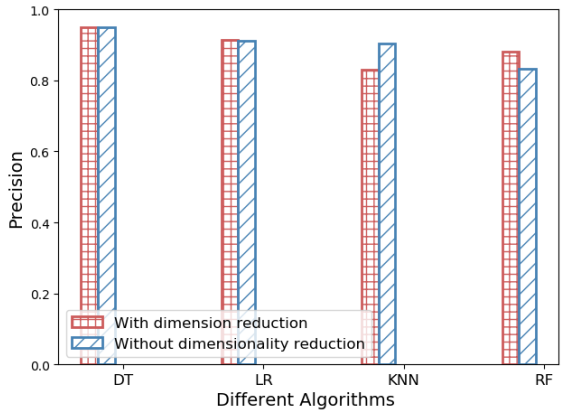

Figure 10. Precision comparison between different algorithms in the dimensionality reduction and non-dimensionality reduction datasets.

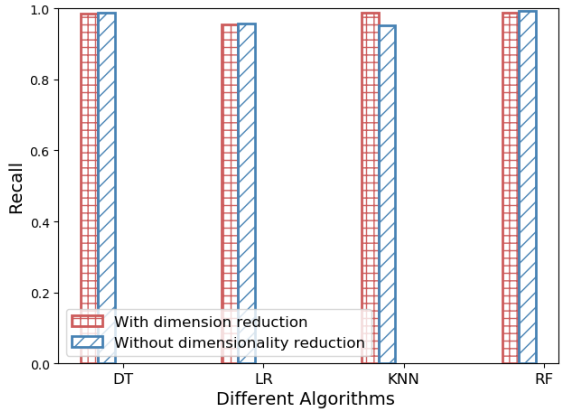

Figure 11. Recall comparison between different algorithms in the dimensionality reduction and non-dimensionality reduction datasets.

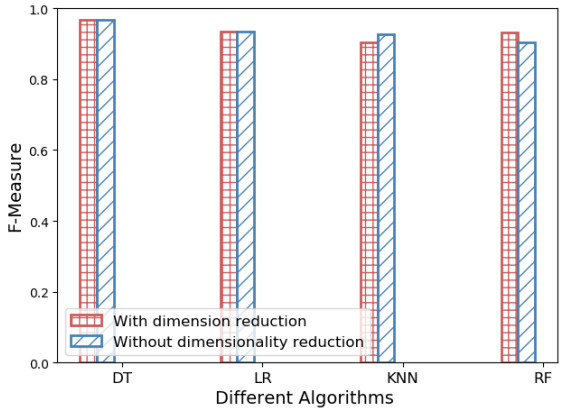

Figure 12. F-measure comparison between different algorithms in the dimensionality reduction and non-dimensionality reduction datasets.

5.3. Model Analysis

The model analysis focuses on verifying whether the detection performance of the capsule network is better than other deep learning networks. We compare the capsule network (CN) with the convolutional neural network (CNN), and the statistical results are shown in Figure 13. In Figure 13, the x-axis involves four different metrics, i.e., accuracy, precision, recall, and F-measure, and the y-axis is the metric value. Compared with CNN, the gaps in the accuracy, precision, recall, and F-measure between CNN and CN are 1.71% 1.41%, 1.07%, and 1.38%, respectively. We further analyzed the factors that produced the experimental results. Figures 14 and 15 show the loss change curve during CN and CNN training, respectively, it can be seen from the loss curve that CN reaches the convergence state faster than CNN. CNN loses some local information through the pooling operation; however, the capsule neural network overcomes this problem, so the robustness of CN is higher than CNN. Through the above analysis, we can conclude that CN is better than CNN. Although the performance of CN is better than that of CNN, CN also has defects considering the time consumption. From the time consumption of the training model, the training time of CN is longer than that of CNN. From the time consumption of the test model, the test time of CN is equivalent to the long test time of CNN.

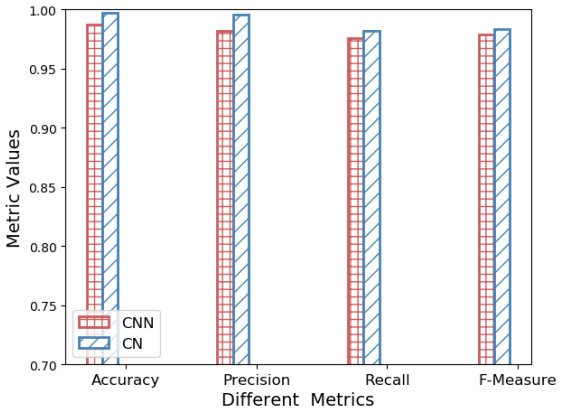

Figure 13. The comparison of CN with CNN.

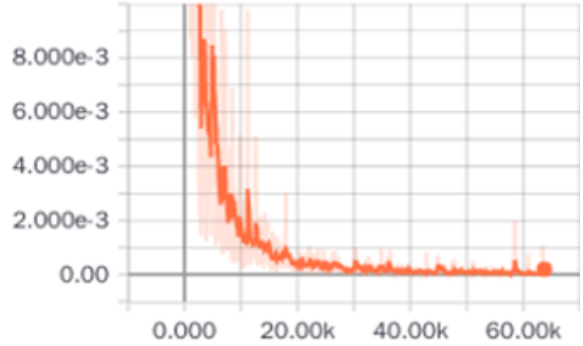

Figure 14. The loss curve of CN.

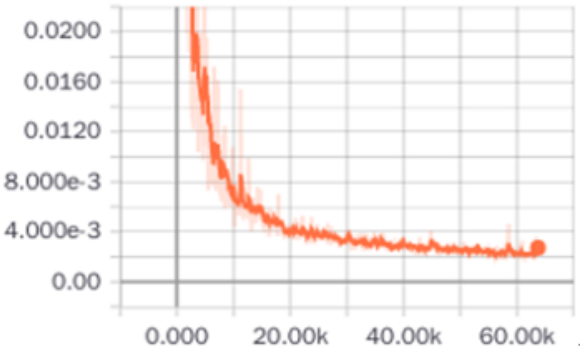

Figure 15. The loss curve of CNN.

5.4. Comprehensive Analysis

We compare our method with the state-of-the-art malware detection technique [9] to further verify the effectiveness of our method. We briefly introduce the principles of this state-of-the-art malware detection technique. The work in reference [9] is a technique that combines supervised and unsupervised learning, and its core lies in the application of the floating centroid method (FCM). The experimental results are shown in Figure 16. Its abscissa is a different metric, and its ordinate is a metric value. From the statistical results, we can conclude that our method is superior to the state-of-the-art malware detection technique in all aspects of performance. The main reason for our experimental phenomenon is that this state-of-the-art malware detection technique is based on simple machine learning algorithms, and our method uses deep learning.

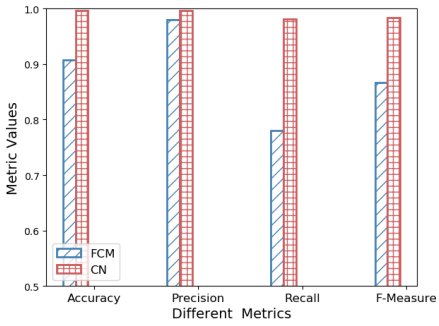

Figure 16. The comparison of CN with other state-of-the-art malware detection techniques.

6. Conclusions

In this work, we propose a feature selection method and apply the capsule network to achieve malware detection. We verify the superiority of our proposed feature selection method and the effectiveness of our application of the capsule network to detect malware from the perspective of feature analysis and model analysis. In feature analysis, we use

four popular machine learning algorithms (decision tree, random forest, logistic regression, and K-nearest neighbor) to train the dataset with feature selection and the dataset without feature selection for the experimental comparison. The experimental results of the feature analysis show that our feature selection method can extract features more conducive to distinguishing benign and malicious samples. In the model analysis, we use convolutional neural networks for comparative experiments. The experimental results of the model analysis show that the capsule network is suitable for network traffic datasets. To further prove the effectiveness of our proposed method, in the comprehensive analysis, we compare it with the malware detection technique presented in [9]. Through the comprehensive analysis of the experimental results, our method provides gains of 9.71% and 20.18% in accuracy and recall, respectively, concerning the results reported in [9]. Although our method performs well, there are still areas for improvement. In future work, we will focus on the adaptive adjustment of parameters in the feature selection method, thereby reducing artificial settings.

Author Contributions: Writing—original draft, K.L. and A.Y.; writing—review and editing, J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Major science and technology project of Hainan Province (Grant No.ZDKJ2020012),Key Research and Development Program of Hainan Province (Grant No.ZDYF2021GXJS003, ZDYF2020040), National Natural Science Foundation of China (NSFC) (Grant No.62162022, 62162024),Hainan Provincial Natural Science Foundation of China (Grant No.621RC1082, 620MS021), the Key Laboratory of PK System Technologies Research of Hainan,Science and Technology Development Center of the Ministry of Education Industry-university-Research Innovation Fund (2021JQR017), Youth Foundation Project of Hainan Natural Science Foundation (621QN211), Beijing Baidu Netcom Science and Technology Co., Ltd. (Grant No.220700001154419).

Data Availability Statement: Not applicable.

Acknowledgments: The Research Group sincerely thanks Hainan University and the Hainan Blockchain Engineering Technology Research Center for the experimental sites and equipment provided for our research.

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

References

1. 2020 State of Malware Report. 2021. Available online: https://securelist.com/ (accessed on 8 January 2023).
2. Wang, S.; Chen, Z.; Zhang, L.; Yan, Q.; Yang, B.; Peng, L.; Jia, Z. TrafficAV: An effective and explainable detection of mobile malware behavior using network traffic. In Proceedings of the 24th IEEE/ACM International Symposium on Quality of Service (IWQoS 2016), Beijing, China, 20–21 June 2016; pp. 1–6. [CrossRef]
3. Liu, B.; Zhang, C.; Gong, G.; Zeng, Y.; Ruan, H.; Zhuge, J. FANS: Fuzzing Android Native System Services via Automated Interface Analysis. In Proceedings of the 29th USENIX Security Symposium, USENIX Security 2020, Boston, MA, USA, 12–14 August 2020; Capkun, S., Roesner, F., Eds.; USENIX Association: Berkeley, CA, USA, 2020; pp. 307–323.
4. Zhang, X.; Wu, K.; Chen, Z.; Zhang, C. MalCaps: A capsule network based model for the malware classification. *Processes* **2021**, *9*, 929. [CrossRef]
5. Omer, M.A.; Zeebaree, S.R.; Sadeeq, M.A.; Salim, B.W.; x Mohsin, S.; Rashid, Z.N.; Haji, L.M. Efficiency of malware detection in android system: A survey. *Asian J. Res. Comput. Sci.* **2021**, *7*, 59–69. [CrossRef]
6. Wei, S.; Zhang, Z.; Li, S.; Jiang, P. Calibrating Network Traffic with One-Dimensional Convolutional Neural Network with Autoencoder and Independent Recurrent Neural Network for Mobile Malware Detection. *Secur. Commun. Netw.* **2021**, *2021*, 6695858 . [CrossRef]
7. Phan, T.V.; Nguyen, T.G.; Dao, N.; Huong, T.T.; Nguyen, H.; Bauschert, T. DeepGuard: Efficient Anomaly Detection in SDN With Fine-Grained Traffic Flow Monitoring. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 1349–1362. [CrossRef]
8. Possemato, A.; Fratantonio, Y. Towards HTTPS Everywhere on Android: We Are Not There Yet. In Proceedings of the 29th USENIX Security Symposium, USENIX Security 2020, Boston, MA, USA, 12–14 August 2020; Capkun, S., Roesner, F., Eds.; USENIX Association: Berkeley, CA, USA, 2020; pp. 343–360.
9. Wang, S.; Yan, Q.; Chen, Z.; Wang, L.; Spolaor, R.; Yang, B.; Conti, M. Lexical Mining of Malicious URLs for Classifying Android malware. In Proceedings of the International Conference on Security and Privacy in Communication Systems, Singapore, 8–10 August 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 248–263.

10. Jagielski, M.; Carlini, N.; Berthelot, D.; Kurakin, A.; Papernot, N. High Accuracy and High Fidelity Extraction of Neural Networks. In Proceedings of the 29th USENIX Security Symposium, USENIX Security 2020, Boston, MA, USA, 12–14 August 2020; Capkun, S., Roesner, F., Eds.; USENIX Association: Berkeley, CA, USA, 2020; pp. 1345–1362.
11. Qiu, J.; Zhang, J.; Luo, W.; Pan, L.; Nepal, S.; Xiang, Y. A survey of Android malware detection with deep neural models. *ACM Comput. Surv. (CSUR)* **2020**, *53*, 1–36. [CrossRef]
12. Abawajy, J.H.; Darem, A.B.; Alhashmi, A. Feature Subset Selection for Malware Detection in Smart IoT Platforms. *Sensors* **2021**, *21*, 1374. [CrossRef] [PubMed]
13. Al-Kasassbeh, M.; Mohammed, S.; Alauthman, M.; Almomani, A. Feature Selection Using a Machine Learning to Classify a Malware. In *Handbook of Computer Networks and Cyber Security, Principles and Paradigms*; Gupta, B.B., Pérez, G.M., Agrawal, D.P., Gupta, D., Eds.; Springer: Berlin/Heidelberg, Germany, 2020; pp. 889–904. [CrossRef]
14. Kishore, P.; Barisal, S.K.; Mohapatra, D.P. JavaScript malware behaviour analysis and detection using sandbox assisted ensemble model. In Proceedings of the 2020 IEEE Region 10 Conference (TENCON 2020), Osaka, Japan, 16–19 November 2020; pp. 864–869. [CrossRef]
15. Yan, A.; Chen, Z.; Spolaor, R.; Tan, S.; Zhao, C.; Peng, L.; Yang, B. Network-based Malware Detection with a Two-tier Architecture for Online Incremental Update. In Proceedings of the 28th IEEE/ACM International Symposium on Quality of Service (IWQoS 2020), Hangzhou, China, 15–17 June 2020; pp. 1–10. [CrossRef]
16. Şahin, D.Ö.; Kural, O.E.; Akleylek, S.; Kılıç, E. A novel permission-based Android malware detection system using feature selection based on linear regression. *Neural Comput. Appl.* **2023**, *35*, 4903–4918. [CrossRef]
17. Fatima, A.; Maurya, R.; Dutta, M.K.; Burget, R.; Masek, J. Android Malware Detection Using Genetic Algorithm based Optimized Feature Selection and Machine Learning. In Proceedings of the 42nd International Conference on Telecommunications and Signal Processing (TSP 2019), Budapest, Hungary, 1–3 July 2019; Herencsar, N., Ed.; pp. 220–223. [CrossRef]
18. Tanuwidjaja, H.C.; Kim, K. Enhancing Malware Detection by Modified Deep Abstraction and Weighted Feature Selection. In Proceedings of the 2020 Symposium on Cryptography and Information Security, Seoul, Republic of Korea, 2–4 December 2020; pp. 1–8.
19. Aminanto, M.E.; Choi, R.; Tanuwidjaja, H.C.; Yoo, P.D.; Kim, K. Deep abstraction and weighted feature selection for Wi-Fi impersonation detection. *IEEE Trans. Inf. Forensics Secur.* **2017**, *13*, 621–636. [CrossRef]
20. Wang, S.; Chen, Z.; Yan, Q.; Ji, K.; Peng, L.; Yang, B.; Conti, M. Deep and broad URL feature mining for android malware detection. *Inf. Sci.* **2020**, *513*, 600–613. [CrossRef]
21. Li, T.; Kou, G.; Peng, Y. Improving malicious URLs detection via feature engineering: Linear and nonlinear space transformation methods. *Inf. Syst.* **2020**, *91*, 101494. [CrossRef]
22. Williams, C.; Seeger, M. Using the Nyström method to speed up kernel machines. In Proceedings of the 14th Annual Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 3–8 December 2001; pp. 682–688.
23. Hokaguchi, T.; Ohsita, Y.; Shibahara, T.; Chiba, D.; Akiyama, M.; Murata, M. Detecting Malware-infected Hosts Using Templates of Multiple HTTP Requests. In Proceedings of the IEEE 17th Annual Consumer Communications & Networking Conference (CCNC 2020), Las Vegas, NV, USA, 10–13 January 2020; pp. 1–2. [CrossRef]
24. Mirsky, Y.; Doitshman, T.; Elovici, Y.; Shabtai, A. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. In Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS 2018), San Diego, CA, USA, 18–21 February 2018.
25. Yan, A.; Chen, Z.; Zhang, H.; Peng, L.; Yan, Q.; Hassan, M.U.; Zhao, C.; Yang, B. Effective detection of mobile malware behavior based on explainable deep neural network. *Neurocomputing* **2020**, *453*, 482–492. [CrossRef]
26. Wang, F.; Wei, Z. A Statistical Trust for Detecting Malicious Nodes in IoT Sensor Networks. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2021**, *104*, 1084–1087. [CrossRef]
27. Cheng, Q.; Wu, C.; Zhou, H.; Kong, D.; Zhang, D.; Xing, J.; Ruan, W. Machine Learning based Malicious Payload Identification in Software-Defined Networking. *arXiv* **2021**, arXiv:2101.00847.
28. Rong, C.; Gou, G.; Cui, M.; Xiong, G.; Li, Z.; Guo, L. MalFinder: An Ensemble Learning-based Framework For Malicious Traffic Detection. In Proceedings of the IEEE Symposium on Computers and Communications (ISCC 2020), Rennes, France, 7–10 July 2020; pp. 1–7. [CrossRef]
29. Chen, Z.; Yan, Q.; Han, H.; Wang, S.; Peng, L.; Wang, L.; Yang, B. Machine learning based mobile malware detection using highly imbalanced network traffic. *Inf. Sci.* **2018**, *433–434*, 346–364. [CrossRef]
30. Sabour, S.; Frosst, N.; Hinton, G.E. Dynamic Routing Between Capsules. In Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017; Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R., Eds.; pp. 3856–3866.
31. Sabour, S.; Frosst, N.; Hinton, G.E. Dynamic routing between capsules. *arXiv* **2017**, arXiv:1710.09829.