Multimedia Tools and Applications (2024) 83:8145–8163
https://doi.org/10.1007/s11042-023-15559-8

# Content-aware malicious webpage detection using convolutional neural network

Yen-Jen Chang[1] · Kun-Lin Tsai[2] · Wei-Cheng Jiang[2] · Meng-Kun Liu[1]

Received: 31 December 2022 / Revised: 7 March 2023 / Accepted: 15 April 2023 /
Published online: 14 June 2023

**Abstract**
Malicious websites often install malware on user devices to gather user information or to disrupt device operations, violate user privacy, or adversely affect company interests. Many commercial tools are available to prevent malicious webpages from accessing devices; however, current versions of these tools may become useless as soon as a new generation of malware is released. In this study, a content-aware malicious webpage detection (CAMD) method was developed; this CAMD method can verify whether a webpage is malicious by applying a novel webpage contextual visualization process, which retrieves the critical codes of webpages, transforms those codes into one-dimensional grayscale images, and applies convolutional neural networks to detect any malicious webpages. To verify the feasibility of proposed CAMD, 50000 normal and 50000 malicious webpages from the VirusTotal website were used. The results indicated that the proposed CAMD achieved an accuracy of $> 98\%$.

**Keywords** Content awareness · Convolutional neural network (CNN) · Malicious webpages · Webpage contextual visualization

## 1 Introduction

Over the past few decades, Internet services have been playing an essential role in our daily lives. Consumers purchase goods from online shopping platforms, such as Amazon

✉ Wei-Cheng Jiang
enjoysea0605@gmail.com

Yen-Jen Chang
ychang@cs.nchu.edu.tw

Kun-Lin Tsai
kltsai@thu.edu.tw

Meng-Kun Liu
whichild@gmail.com

[1] Department of Computer Science and Engineering, National Chung Hsing University, Taichung City, Taiwan

[2] Department of Electrical Engineering, Tunghai University, Taichung City, Taiwan

and eBay. Social media sites such as Facebook and Twitter allow friends to communicate with friends and allow fans to follow idols. Moreover, mobile apps and webpages often give access to useful Internet services. Some web designers seek to enhance webpages with embedded syntax and background processes. However, malicious functions in webpages can steal user information [7, 27].

Numerous types of malicious webpages currently exist, including drive-by downloads, clickjacking, phishing, social engineering attacks, and so on [24]. In drive-by downloads, users unintentionally download malware [8]. In clickjacking, a user is tricked into clicking on something different from what they perceive; thus, confidential information can be revealed [14]. Phishing attacks use e-mail or interactive webpages to steal identification [2]. Social engineering attacks trick people by using social relationships to swindle users into revealing valuable information [13]. Identifying the aforementioned attacks is difficult for many users when they access webpages by using Internet services, and these attacks often cause considerable damage [34]. Thus, an effective method to detect malicious webpages substantially helps people securely browse webpages.

In this paper, a content-aware malicious webpage detection (CAMD) method is proposed to examine webpage syntax through webpage contextual visualization. CAMD can be used to detect not only HTML style languages but also JavaScript and cascading style sheets. Natural language processing systems can utilize word2vec models [12] in their learning procedures, but programs to analyze webpage syntax confront much more complex problems and thus cannot simply use word2vec. To resolve the problem of complicated and varied webpages, the proposed CAMD uses a webpage contextual visualization as a preprocessing procedure, which retrieves the critical codes of webpages according to the Token-ASCII-Sum approach. Retrieved codes were transformed into one-dimensional grayscale images and then were entered as inputs into a convolutional neural network (CNN) to distinguish between good and bad. According to the experimental results, the CAMD provided 98.08% of predictive accuracy; the area under curve (AUC) value was 0.995 with a 2.598% false positive rate (FPR). Moreover, CAMD exhibited $>$ 98% true positive rate (TPR) performance. The experimental results indicated that the proposed method outperformed previous approaches.

The rest of this paper is organized as follows. Section II presents related studies. Section III provides a brief introduction of deep learning and CNN concepts. Section IV provides the description of the CNN-based CAMD method. Section V presents the results of experiments performed to validate the proposed method. Section VI presents the conclusion and details of opportunities for future studies.

## 2 Related studies

The detection of malicious webpages presents two challenges. First, the computational complexity of a detection method should be as low as possible to maintain the browser's performance during malicious webpage detection. Second, attack techniques evolve daily; thus, malicious webpage detection should be flexible and frequently updated. The blacklist approach used by many antivirus companies is a common method for detecting malicious webpages [11, 31]. However, the blacklist approach is hampered by many limitations. For example, a new malicious webpage can easily escape when the blacklist approach is used for detection; thus, the blacklist approach cannot overcome the second challenge. To overcome such a limitation, machine learning algorithms have been introduced for detecting malicious webpages [24, 26, 27]. Unlike the blacklist approach, machine learning algorithms exhibit an ability to detect new malicious webpages. These methods may not achieve higher accuracy, but they can be effectively selected suspicious malicious webpages. Therefore, these methods can be used to choose suspicious malicious webpages for other high-precision method that can conduct further test. Therefore, some studies have proposed machine learning to prevent the attacks of malicious webpages [1, 6, 10, 25, 26, 28, 29, 33, 37]. In [28], an automatic feature extraction method, in which a character-level embedding was combined with a CNN, was proposed, and some fragmented Uniform Resource Locator (URL) data were used as input data to detect malicious webpages. However, the feature extraction procedure of this neural network was complicated. Canali et al. [6] proposed a webpage filter, Prophiler. It retrieved various features from HyperText markup language (HTML), JavaScript, and URLs for the detection of malicious webpages. Their results indicated that Prophiler performed efficiently with little computational complexity. However, detecting malicious webpages by using the aforementioned methods in a short time is difficult, and these methods often require many computing resources. Moreover, the format of webpage features must be defined when using a parser. Therefore, it is not flexible in practical applications. Abdi et al. [1] detected malicious webpages by employing a multilayer architecture, in which the first layer was similar to the blacklist approach. A URL was compared with an existing blacklist, and the matched URL was directly judged as the malicious webpage. When the URL passed the first layer, a term frequency-inverse document frequency (TF-IDF) statistical method and word2vec neural network were used to extract a URL feature. Subsequently, the CNN, a support vector machine, and linear regression were used to determine whether the URL was malicious. This method can be useful for detecting malicious URLs, but the blacklist must be maintained and updated frequently. Additionally, some URLs are exploited to redirect to malicious webpages due to their forwarding mechanism. However, to acquire user's security information, malicious websites usually disguise itself as normal websites. To detect the abnormal attacks hidden in the malicious websites, various deep learning models are efficient for the malicious webpage detection [21, 35, 36]. In [36], a convolutional gated-recurrent-unit (GRU) neural network is proposed for malicious Uniform Resource Locators (URLs) detection. The URLs are composed of characters which are used to retrieve the text features for classification. A feature representation views that the malicious keywords are unique for the URLs, so it will use the malicious keywords to form a representation for the (GRU) model as input. The results show that the feature representation method helps the (GRU) model obtain high accuracy. In [35], it proposes an algorithm, called URL embedding (UE), which is used to find coefficients of URLs. The method uses a distributed feature representation for URLs to avoid the curse of dimensionality. Using the UE method, it will obtain a low-dimensional vector which includes important feature information and helps deep learning model obtain high accuracy for malicious attack detection. In [21], it proposes a malicious website detection method from user's perspective for detecting web spams. The method based on a Convolutional Neural Network uses as a classification algorithm and it also discusses various Web spam techniques. In [5], it introduces a MSCA-net to integrates Multi-Scale Contextual Feature Fusion and Multi-Context Attentional Feature. The net captures contextual information to accurate the image segmentation. The model can improve the conventional segmentation method for decision-making. Moreover, many crucial deep learning models, such as LeNet [19], AlexNet [18], GoogLeNet [32], and VGGNet [30] are used in different fields, such as game [23], internet [9] and medical applications [3, 4] to perform classification and recognition applications. These famous models provide excellent performance. However, the proposed CAMD model extracts webpage syntax using HTML, CSS and Javascript code, through webpage contextual visualization. The Token-ASCII-Sum method extracts codes from original webpage and transforms into one-dimensional grayscale images which is from the feature representation. Because the proposed model is an approach of the content-aware malicious webpage detection. When the model meets a malicious webpage and the order of the codes in the malicious webpage is reorganized to a harmless webpage. The proposed model cannot be influenced by the reorganization and can also classify the webpage to normal or malicious one. Therefore, in the experimental section, the model can be implemented in real situation.

## 3 Content-aware malicious webpage detection using CNN

The CNN is also used to distinguish between normal and malicious webpages. The problem of malicious webpage detection can be considered a binary classification problem, and the detection result should be normal or malicious webpages. Accordingly, the problem of malicious webpage detection is defined as follows.

A data set $D$ contains various original webpages, $D = \left\{(t_1, y_1), (t_2, y_2), ..., (t_i, y_i)\right\}$, where $i = 1, 2, 3, ..., |D|$; $|D|$ is the cardinality of the data set; and $y_i \in \left\{0, 1\right\}$. $y_i = 0$ represents that $t_i$ is a malicious webpage, and $y_i = 1$ represents that $t_i$ is a normal webpage.

Figure 1 reveals that the CAMD architecture includes three parts: (A) data preprocessing, (B) WcvNet model training, and (C) WcvNet model testing.

### 3.1 Data preprocessing

Data was preprocessed to retrieve critical codes from a webpage under test to a visible vector, which can be further represented as a grayscale image. The preprocessing procedure, Token-ASCII-Sum, comprises two steps. The first step converts the critical code vector $W$

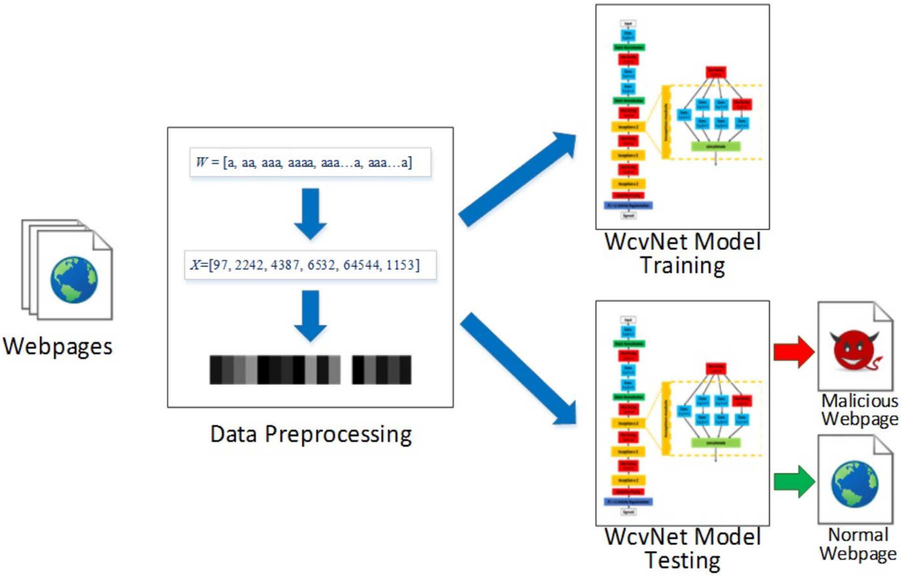


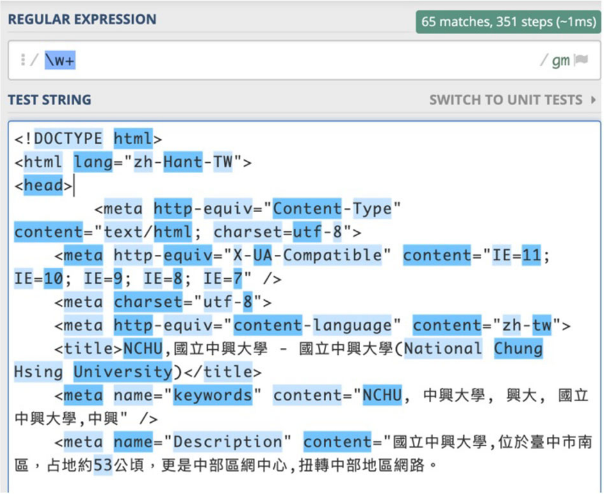
**Fig. 1** CAMD architecture

of an under test webpage $t_i$ into a numerical vector $X$. Subsequently, the numerical vector $X$ was transformed into a visualized vector $V$ to filter noises and increase the prediction accuracy.

In the first step, a regular expression was employed for word string comparison. Figure 2 reveals that a regular expression instruction [\w+] was applied to retrieve the webpage content, and the results are marked as blue blocks. By using the instruction [\w+], English uppercase and lowercase alphabets (a-z and A-Z), numbers (0–9) and underscores ( _ ) from the original webpages were retrieved, and the remainder, for example Chinese characters and symbols, were ignored. Figure 2 presents an example wherein a total of 65 tokens (strings), that is, $w_i$ and $i = 1, 2, ..., 65$, are marked. All 65 $w_i s$ formed a token vector $W$, where $W = \left\{w_i, \forall i\right\}$. Subsequently, each $w_i s$ was transferred into a corresponding value $x_i$, and all $x_i s$ formed the numerical vector $X$, that is, $X = \left\{x_i, \forall i\right\}$. The Token-ASCII-Sum transferring is presented as (1).

$$x_i = \left((|w_i| - 1) \mod N \right) \times \lceil \tfrac{65535}{N} \rceil + \left(\sum_{CT=1}^{|w_i|} A_{CT}\right) \mod \lceil \tfrac{65535}{N} \rceil \tag{1}$$

where $|w_i|$ is the length of token $w_i$; $N$ represents the pre-defined constant number represents the predefined constant number, which is used to partition a long token; $CT$ is the ordinal number of character of token $w_i$; $A_{CT}$ denotes the corresponding ASCII code of each $CT$; and the number 65536 is the numerical range of 16 bits grayscale image. In our proposed method, when the Token-ASCII-Sum process uses fewer bits to represent grayscale images, some important features may be ignored in the transformation from original webpages to grayscale images. In other words, the grayscale image cannot keep sufficient feature information. On the contrary, the more bits used for representing the grayscale, the higher accuracy of classification can be obtained; however, it needs more hidden layers for the proposed model and results in performance



```
/ \w+ / gm
```

```
<!DOCTYPE html>
<html lang="zh-Hant-TW">
<head>
        <meta http-equiv="Content-Type"
content="text/html; charset=utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=11;
IE=10; IE=9; IE=8; IE=7" />
    <meta charset="utf-8">
    <meta http-equiv="content-language" content="zh-tw">
    <title>NCHU,國立中興大學 - 國立中興大學(National Chung
Hsing University)</title>
    <meta name="keywords" content="NCHU, 中興大學, 興大, 國立
中興大學,中興" />
    <meta name="Description" content="國立中興大學,位於臺中市南
區，占地約53公頃，更是中部區網中心,扭轉中部地區網路。
```

**Fig. 2** Example of word string comparison by using regular expression

penalty. In real situation, the end-to-end predication of model must cost much time for the recall process. So, the 16 bits (0-65535) are chosen for the Token-ASCII-Sum process.

Considering the string “holder” an example, the parameters of the string were set as $|w_i| = 6$, $CT = 1$, and $N = 4$. The ASCII code of each character was as follows: h = 104, o = 111, l = 108, d = 100, e = 101, and r = 114; consequently, the summation of $A_{CT}$ of this string was 638. According to (1), the $x_i$ of token “holder” was 17022. When the token vector $W$ = “This, is, a, pen, holder” was retrieved from a certain webpage, the corresponding numerical vector was $X$ = [49560, 16604, 97, 33091, 17022], which is a 5D vector, and each element $x_i \in \mathbb{R}^5$ is a scalar. The $N$ value affected the accuracy of CAMD final decision. In the second step of data preprocessing, the numerical vector $X$ was transformed into a visualized vector $V$.

Consider another token $W$ = [a, aa, aaa, aaaa, aaa…a$^1$, aaa…a$^2$] an example, where aaa…a$^1$ exhibits 32 continuous characters ’a’ and aaa…a$^2$ exhibits 33 continuous characters ’a’. When $N$ was set to 32 by following the process presented in the first step, the corresponding numerical vector $X$ was [97, 2242, 4387, 6532, 64544, 1153]. To obtain the visualized vector of webpage $t_i$, each $x_i$ of $X$ was mapped to a grayscale level image. Figure 3 illustrates that the grayscale level was set from 0–65535 (the last value is equal to $2^{16} - 1$). The mapping result was used to form a visualized vector $V = \left\{v_i, \forall x_i \in X \text{ and } x_i \rightarrow v_i\right\}$. Figure 3 presents the mapping result of $W$. When $x_i$ was small, the Binarization of the grayscale image was close to black color; otherwise, it was close to white color.

Figure 4 presents two examples of transformation from a source webpage to a visualized vector. Figure 4(a) is a example of a malicious webpage. Figure 4(b) is a example of a harmless webpage. This visualized vector was a 1 × 17 vector represented by a grayscale image, which was also the input data of the proposed deep neural network. One webpage under test was converted to a grayscale image to retain the spatiality of the original text of the webpage. However, the sizes of token vectors retrieved from different webpages varied, which complicated the CNN training process. In this study, the token vector size

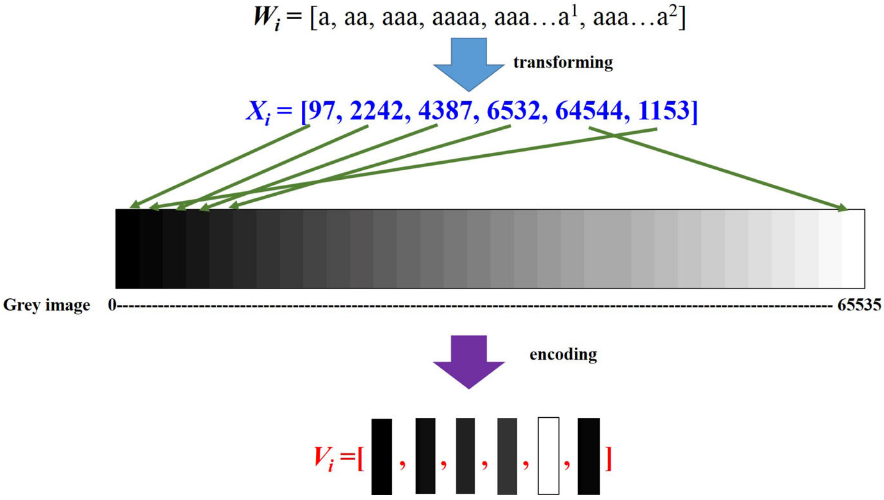
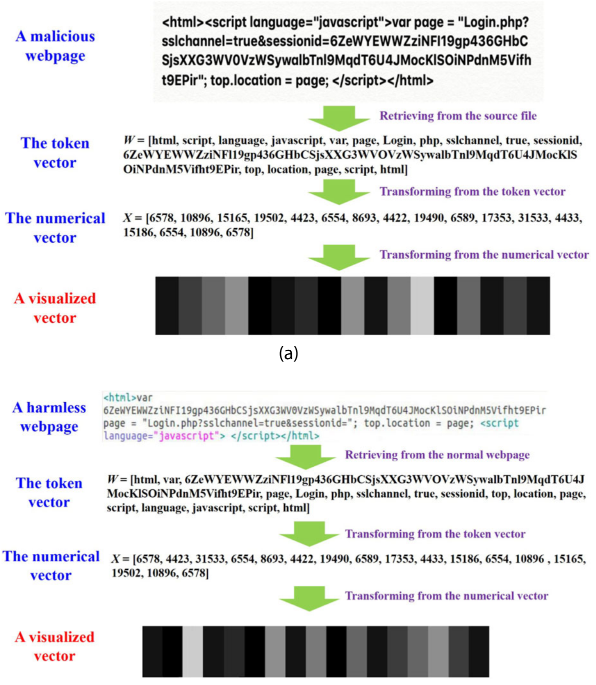

$W_i$ = [a, aa, aaa, aaaa, aaa…a$^1$, aaa…a$^2$]
transforming
$X_i$ = [97, 2242, 4387, 6532, 64544, 1153]
Grey image
0
65535
encoding
$V_i$ =[ , , , , , ]

**Fig. 3** Process of the transferring the visible vector


A malicious webpage
Retrieving from the source file
The token vector
Transforming from the token vector
The numerical vector
Transforming from the numerical vector
A visualized vector
A harmless webpage
Retrieving from the normal webpage
The token vector
Transforming from the token vector
The numerical vector
Transforming from the numerical vector
A visualized vector

```
<html><script language="javascript">var page = "Login.php?
sslchannel=true&sessionid=6ZeWYEWWZziNFl19gp436GHbC
SjsXXG3WV0VzWSywalbTnl9MqdT6U4JMocKlSOiNPdnM5Vifh
t9EPir"; top.location = page; </script></html>
```

*W* = [html, script, language, javascript, var, page, Login, php, sslchannel, true, sessionid, 6ZeWYEWWZziNFl19gp436GHbCSjsXXG3WVOVzWSywalbTnl9MqdT6U4JMocKlSOiNPdnM5Vifht9EPir, top, location, page, script, html]

*X* = [6578, 10896, 15165, 19502, 4423, 6554, 8693, 4422, 19490, 6589, 17353, 31533, 4433, 15186, 6554, 10896, 6578]

(a)

```
<html>var
6ZeWYEWWZziNFI19gp436GHbCSjsXXG3WV0VzWSywalbTnl9MqdT6U4JMocKlSOiNPdnM5Vifht9EPir
page = "Login.php?sslchannel=true&sessionid="; top.location = page; <script
language="javascript"> </script></html>
```

*W* = [html, var, 6ZeWYEWWZziNFl19gp436GHbCSjsXXG3WVOVzWSywalbTnl9MqdT6U4JMocKlSOiNPdnM5Vifht9EPir, page, Login, php, sslchannel, true, sessionid, top, location, page, script, language, javascript, script, html]

*X* = [6578, 4423, 31533, 6554, 8693, 4422, 19490, 6589, 17353, 4433, 15186, 6554, 10896 , 15165, 19502, 10896, 6578]

(b)

**Fig. 4** Example of transformation of a source file of a webpage into a visualized vector. (a) An example of a malicious webpage (b) An example of a harmless webpage

was unified as 1 × 14400, because the maximum token size retrieved from webpages by applying regular expression instruction [\w+] was 14339. After the transformation of the token vector into a numerical vector, if the size of the numerical vector was smaller than 1 × 14400, the insufficient parts of the numerical vector were added by zeros, which is zero-padding.

### 3.2 WcvNet model training

To classify malicious and normal webpages, a novel WcvNet neural network was proposed, which was a GoogLeNet-based CNN model. GoogLeNet, which was proposed by Szegedy et al. [32] in 2014, enhances the network width and uses different sizes such as 1 × 1, 3 × 3, and 5 × 5 for extracting features. The 1 × 1 convolution operation could reduce the dimension of feature maps to avoid large parameters after deepening and widening of networks.

All convolution layers and pooling layers in the WcvNet were designed specifically for training 1D grayscale images, where the feature detector used in convolution layers exhibited the smallest size of 1 pixel and the largest size of 5 pixels. By matching the 1-pixel stride, the continuous syntax fragment in web source files was extracted to further interpret the syntax fragments of web source files with different lengths. The 3-pixel pooling kernel used in the pooling layer could concentrate features from the convolution feature map of the previous layer and retain the adjacent relation between the context of the web source file. When the model entered the inception layer, the pooling stride was set to 2 to highly reduce

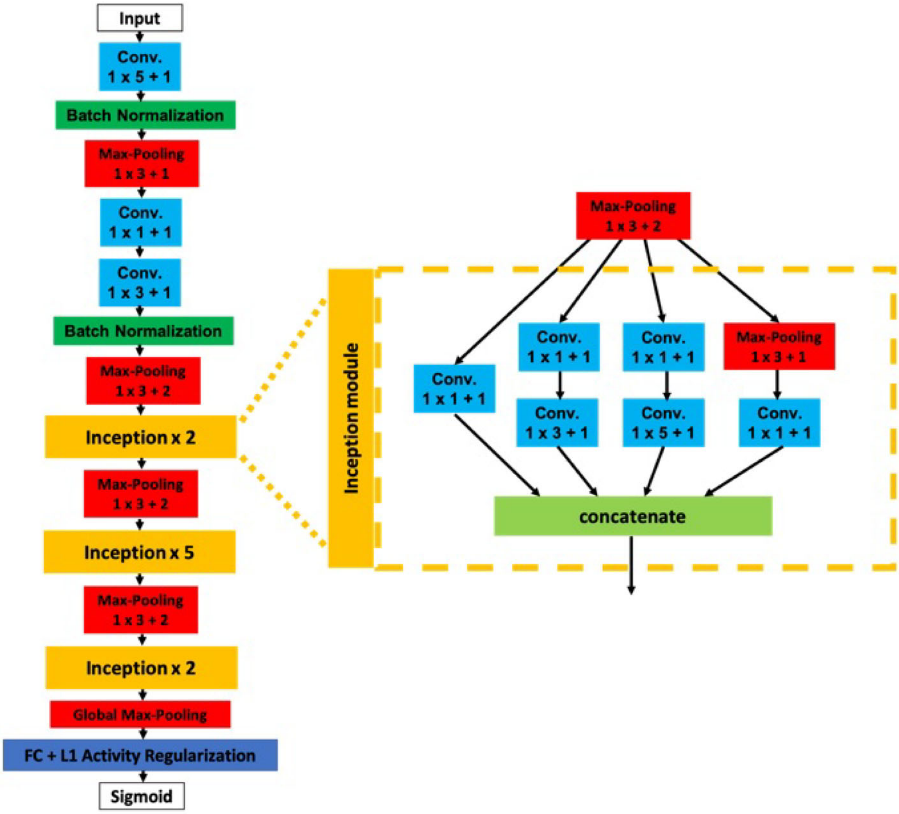


**Fig. 5** Architecture of the proposed WcvNet network

operation parameters and balance the accuracy and effectiveness. Figure 5 illustrates the WcvNet architecture, where conv. $1 \times 5 + 1$ is the feature detector of size $1 \times 5$ with one-pixel stride used in convolution layer, and max-pooling $1 \times 3 + 1$ represents the pooling kernel size of $1 \times 3$ with one-pixel stride.

In the activation function of WcvNet, ReLU was used for all hidden layers and a sigmoid was used in the output layer. Convolutional layer, activation function, fully connected layer, and three optimal elements, namely batch normalization, global max-pooling, and activity regularization, were employed in WcvNet.

### 3.2.1 Batch Normalization (BN)

*BN* is used in the WcvNet network for processing the input data to ensure that the data remains in an effective range when it is transmitted from one layer to another. The advantage of *BN* is to maintain the distribution of original data; thus, it is a method for optimizing neural networks. *BN* is often introduced after the convolution layer or fully connected layer. This method divides the data into small batches for stochastic gradient descent. When each separated batch of data was in forward propagation, each layer was normalized as illustrated in Fig. 6. A min-batch, $B = \{z_1, z_2, ..., z_j\}$, where $z_j$ represents a weight value of a neuron

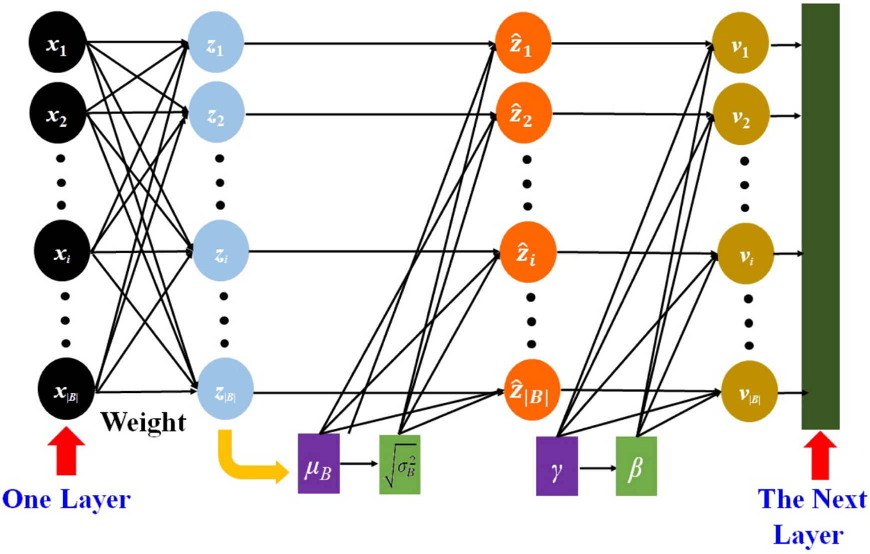


**Fig. 6** Example of batch normalization from one layer to another layer

and $j = 1, 2, ..., |B|$. During training, each mini-batch must be used to calculate the mini-batch mean, $\mu_B$, and mini-batch variance, $\sigma_B^2$, as follows.

$$\mu_B = \frac{1}{|B|}\sum_{j=1}^{|B|} z_j \tag{2}$$

$$\sigma_B^2 = \frac{1}{|B|}\sum_{j=1}^{|B|} z_j^2 - \mu_B^2 \tag{3}$$

where $\sqrt{\sigma_B^2}$ is the standard deviation of the variance, $\sigma_B^2$.

When the mini-batch mean and mini-batch variance were obtained, the value, $z_j$, was used for normalization to form a value, $\hat{z}_j$, as follows.

$$\hat{z}_j = \frac{z_j - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{4}$$

where $\epsilon$ is an arbitrarily small constant, which is added in the denominator to avoid the problem of dividing by zero.

The obtained normalized value, $\hat{z}_j$, can directly entered as an input into the activation function. The normalized value was further scale and shift listed as follows.

$$v_j = \alpha\hat{z}_j + \gamma \tag{5}$$

where $\alpha$ and $\gamma$ are the translation and extended parameters, respectively. The neural network used them to scale and modify the normalized value, $\hat{z}_j$. By using these two parameters, the WcvNet network slowly realized whether the normalization operation achieved optimization. When the normalization operation did not work well, the two parameters can cancel out the operation [15]. When the network experiences slow convergence, gradient vanishing, and gradient exploding, the *BN* method can be introduced to resolve these problems.

#### 3.2.2 Global Max-Pooling (GMP)

In original fully connected layers of CNN as described in Section III, the *GMP* method [20] is employed to determine the maximum value of feature map generated by the previous convolution layer, to reduce the dimension, and to prevent overfitting. For using the GMP method, considering the number of neurons required by the fully connected layer is not necessary.

#### 3.2.3 L1 activity regularization

WcvNet is used to decrease the error of the loss function $L$, which is given as follows.

$$L = (d - v)^2 \tag{6}$$

where $d$ and $v$ represent the desired output and output of the network, respectively.

Generally, the loss function is used to evaluate the difference between the desire output and output of the neural network. L1 regularization is used to add the summation of the absolute values of weight as a penalty term to the loss function. The loss function using L1 regularization becomes smoother and reduces the noise influence. The penalty term is given as follows.

$$L = (d - v)^2 + \lambda \sum |Weights| \tag{7}$$

where $\lambda$ is constant used to control the regularization strength. The penalty term enabled the neural network to learn sparse features and prevent overfitting. WcvNet employed L1 activity regularization for the fully connected layer.

After the completion of the sigmoid function, WcvNet exhibited only one output value, which was located in the interval (0, 1). An output value larger than a given threshold indicated the malicious webpage, whereas the output value smaller than the given threshold suggested the normal webpage.

### 3.3 WcvNet model testing

To detect malicious webpages, the proposed WcvNet model was used to minimize the loss function and to increase the prediction accuracy. During model testing, a testing data set that was not to be used for model training are employed to evaluate the loss value of the WcvNet model.

To obtain the superior classification result, an adaptive learning rate tuning method was employed in the testing phase to reduce the loss value by few epochs. The adaptive learning rate tuning method was used to set an initial learning rate for renewing the weights of all neurons, and then, in each epoch, the loss value was calculated and recorded. For loss value optimization, adaptive moment estimation (Adam) [16] was used. Adam, a self-adaptive learning rate algorithm, was employed to maintain the learning rate of each epoch in a stable parameter range after the correction of training parameters. Therefore, Adam was used to rapidly decreased the loss value and can often be utilized to optimize the CNN.

## 4 Experiment

Training and verification data sets used in this study were obtained from the network threat intelligence website, VirusTotal [17, 33]. VirusTotal receives numerous HTML files every day and judges the malicious content by using 60 network threat scanning mechanisms

**Table 1** Dataset structure of this study

| | Malicious webpages | Normal webpages |
|---|---|---|
| Training data | 45,000 | 45,000 |
| Testing data | 5,000 | 5,000 |

offered by 12 information security businesses. As a preventive measure , the acquired data sets were scanned with an antivirus software in personal computers to determine whether the source data was normal or malicious before the experiment. Table 1 presents the quantity and distribution proportion of data sets. A total of 100,000 webpages were adopted, including 50,000 malicious webpages and 50,000 normal webpages.

### 4.1 Comparison between the different settings of parameters

In model training, a large amount of training data restricted to hardware operation resources cannot be loaded simultaneously . Generally, for training, the entire data sets are generally input to the model as batches. The batch size used for the training model was 20, the iteration was 1000, and the epoch was 40. Such a distribution, under the lab environment and the highest operation resource utilization provided the most favorable balance for the model training effectiveness.

To obtain the accurate classification result from the WcvNet, reducing the loss value down to the lowest in training is essential. Figure 7 presents the learning curve of the loss value, where the $x$-axis and $y$-axis represent the epoch and loss value, respectively. The variation of the loss value is recorded in the Fig. 7. In the $1^{st}$ epoch, the error is approximately 0.371 and the accuracy approximately reaches 86%. In the $38^{th}$ epoch, the error lowest loss value was 0.126 and the accuracy can reach 98.1%. However, from $22^{nd}$ to $37^{th}$ epoch, the accuracy approximately reaches 97%. The detailed information of error and accuracy with different epochs are recorded in Figs. 7 and 8.

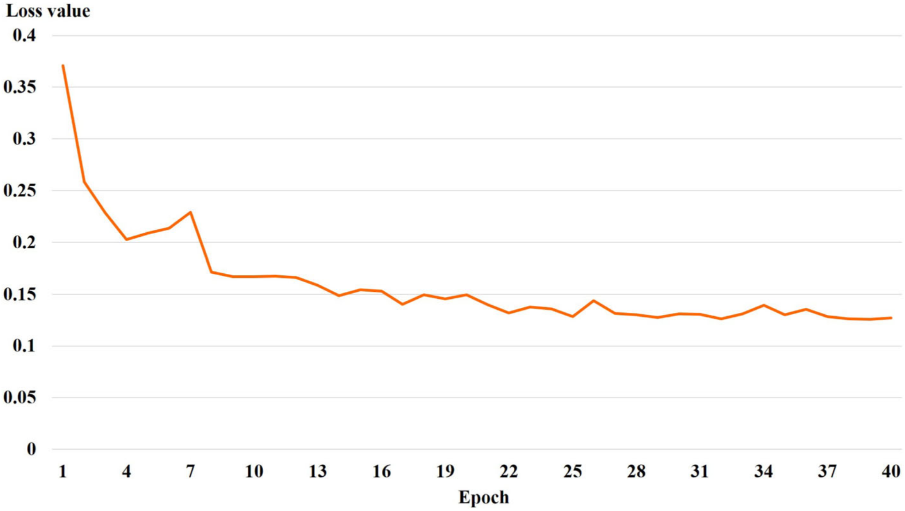


**Fig. 7** Learning curve of the loss value

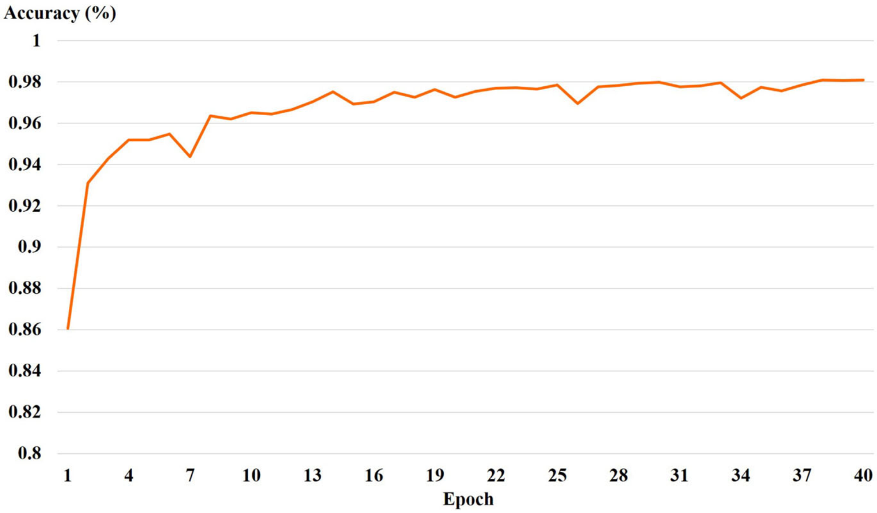


**Fig. 8** Accuracy of WcvNet

When the loss value of the network did not decrease after two epochs, then a half of the current learning rate was automatically reduced to properly converge the loss value. After model training and verification with several parameter combinations, the superior initial learning rate was 0.0001, the loss automatic reduction tolerance was 2, and the loss automatic reduction ratio was 50%.

The $N$ value influences the accuracy of CAMD's final decision. To investigate the effect of parameter $N$, separated tokens with different $N$ were compared, and the results are presented in Fig. 9, where the $x$-axis and $y$-axis represent the epoch and prediction accuracy,

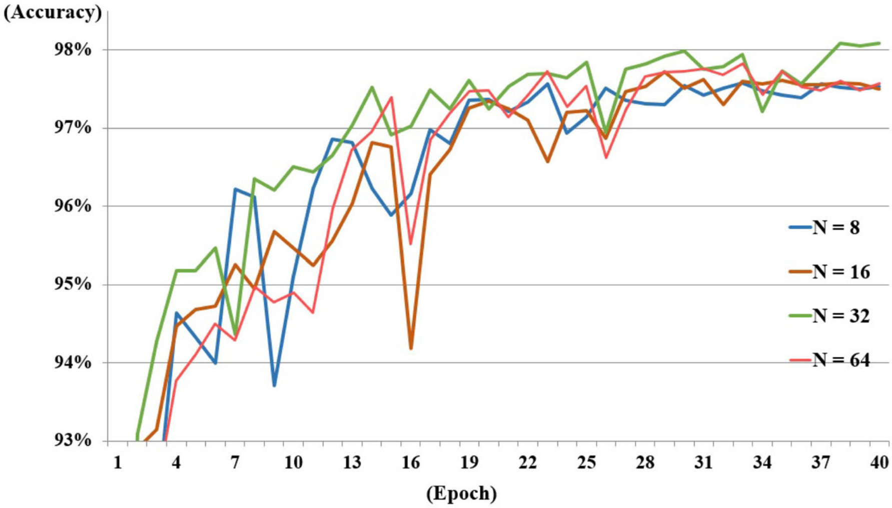


**Fig. 9** Comparison of separated tokens with different $N$

**Table 2** Numbers of parameters used for three models

| Models | Number of parameters |
|---|---|
| AlexNet | 119,911,425 |
| LeNet | 27,668,865 |
| HIA | 4,211,713 |
| CAMD | 3,442,393 |

respectively. Figure 9 indicates that when $N = 32$, the proposed CAMD obtained the highest accuracy; thus, parameter $N$ was set to 32 in the subsequent experiments.

### 4.2 Comparison of the CAMD Architecture with Previous Methods

Three crucial deep learning models, LeNet [19], AlexNet [18] and hierarchical inspector approach (HIA) [29], were used to evaluate the accuracy of malicious webpage decision and were compared with the proposed CAMD. The same data preprocessing phase was used for LeNet and AlexNet, and the same grayscale images were inputted in these models. The HIA model uses regular expression to retrieve source webpages to form a token stream which is divided into 16 equal parts. Then, it executes the hashing trick for each division and uses hierarchical features as input data for the processes of feature extraction and classification.

**Table 3** Confusion Matrices of Four Methods: (a) CAMD, (b) HIA, (c) LeNet, and (d) AlexNet

| | | | |
|---|---|---|---|
| (a) | | | |
| CAMD | | Predicted | |
| | | Positive | Negative |
| Actual | Positive | TP=4931 | FN=69 |
| | Negative | FP=123 | TN=4877 |
| (b) | | | |
| HIA | | Predicted | |
| | | Positive | Negative |
| Actual | Positive | TP=4945 | FN=55 |
| | Negative | FP=254 | TN=4746 |
| (c) | | | |
| LeNet | | Predicted | |
| | | Positive | Negative |
| Actual | Positive | TP=4065 | FN=935 |
| | Negative | FP=557 | TN=4443 |
| AlexNet | | Predicted | |
| | | Positive | Negative |
| Actual | Positive | TP=4774 | FN=226 |
| | Negative | FP=372 | TN=4628 |

Table 2 presents the total numbers of parameters used in the CAMD, LeNet, and AlexNet, and the proposed model exhibits the lowest number of the parameters.

In model training, the training data with labels were entered as inputs to train the network and adjusts the neuron weights. The validation data were used to test the classifier. The CAMD is a binary classifier used for the malicious and normal webpage determination. The threshold for the binary classification of malicious and normal webs was 0.5; a value > 0.5 was considered positive (positive/malicious web) and that < 0.5 was considered negative (negative/normal web). True positive (TP), false negative (FN), true negative (TN), false positive (FP), and commonly used extended evaluation standards in the confusion matrix for binary classification were used for evaluating the model reliability. A total of 10,000 pieces of verification data were entered as input in CAMD, HIA, LeNet, and AlexNet. Table 3 presents the confusion matrices.

After the generation of the confusion matrices of model evaluation, the indicators, including TP, FN, TN, FP, accuracy, precision, recall, F1-score, true positive rate (TPR), and false positive rate (FPR), were calculated to evaluate the accuracy of the deep learning model. The relevant evaluation indicators are given as follows.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

$$\begin{aligned} &F_{\beta-score} = \quad (1+\beta^2)\tfrac{Precision \times Recall}{\beta^2 \times Precision + Recall} \\ &if \quad \beta = 1 \\ &\qquad F_{1-score} = 2 \times \tfrac{Precision \times Recall}{Precision + Recall} \end{aligned} \tag{11}$$

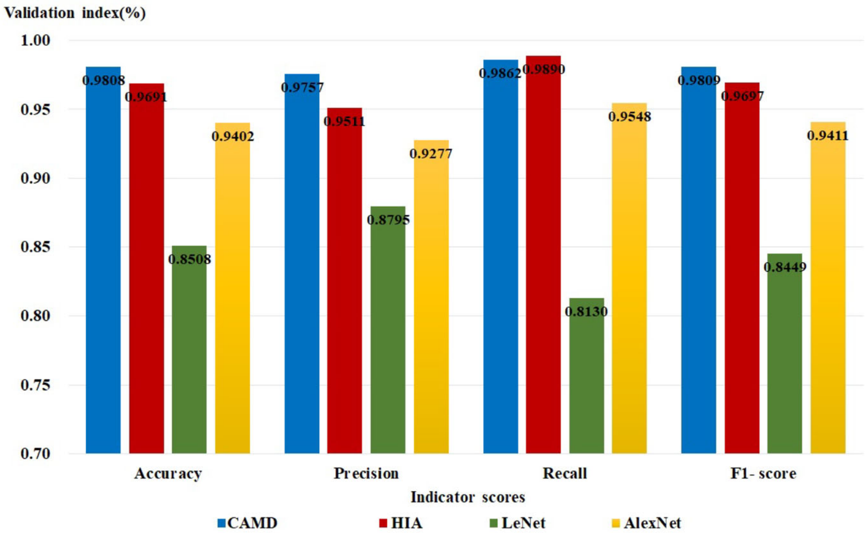


**Fig. 10** Validation indices for each method

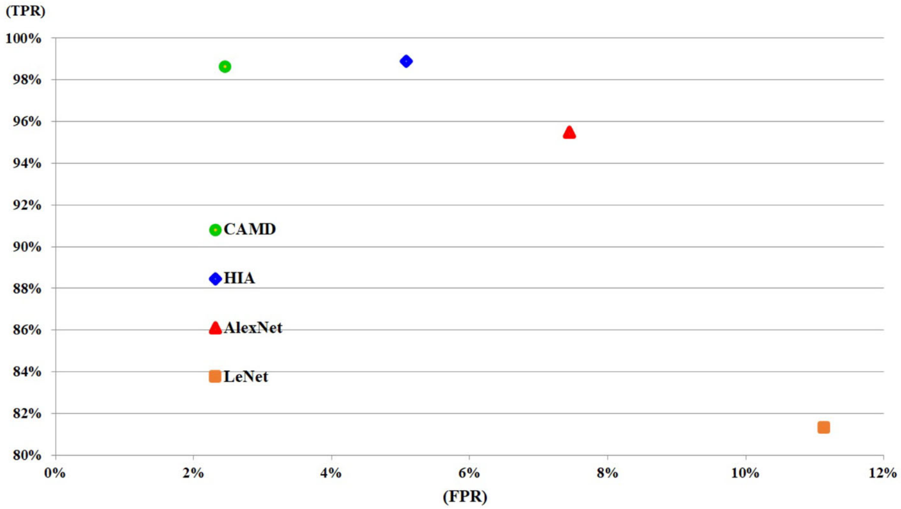


**Fig. 11** TPR and FPR indices for each method

$$FPR = \frac{FP}{TN + FP} \tag{12}$$

where TPR is equal to Recall.

Figure 10 presents the model evaluation data obtained using these equations, where the $x$-axis and $y$-axis represent indicators and the validation index percentage, respectively. Figure 10 illustrates that the proposed CAMD provided the accurate evaluation of accuracy, precision, and F1-score. In addition, the validation indices of TPR and FPR were calculated, as presented in Fig. 11, where the $x$-axis and $y$-axis represent the index of FPR and TPR,

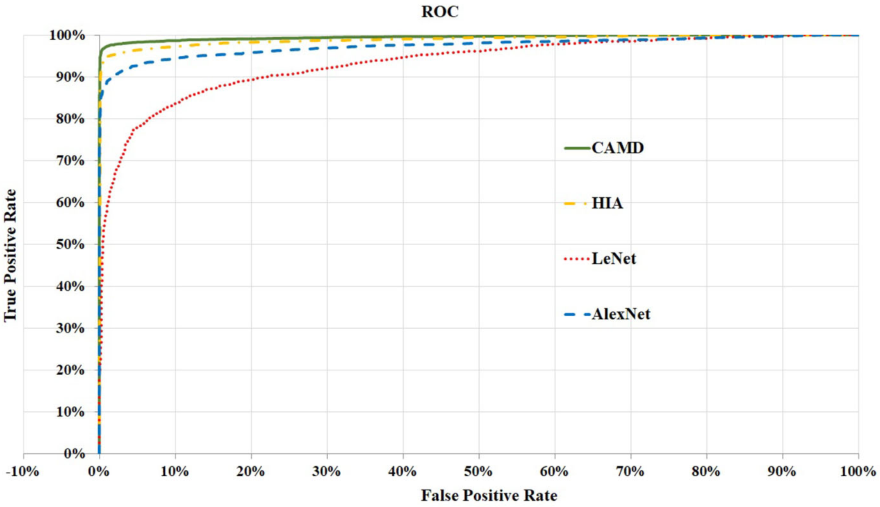


**Fig. 12** ROC curves

**Table 4** AUC values of the four methods

| | AUC |
|---|---|
| CAMD | 0.995 |
| HIA | 0.989 |
| LeNet | 0.934 |
| AlexNet | 0.974 |

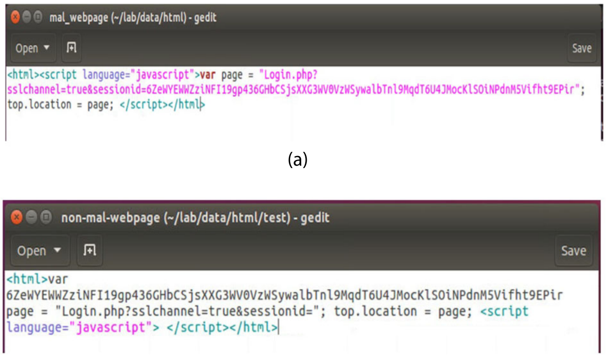
respectively. When the CAMD was 2.5% FPR, it achieved 98% TPR. The proposed CAMD achieved high predictive accuracy.

The evaluation data were measured based on a classification threshold of 0.5. To evaluate the performance, a receiver operating characteristic (ROC) curve was employed with different thresholds were used for binary classification as the measurement benchmark. The ROC curve was closer to the upper left corner to reveal the fixed cost (FPR) corresponding to the benefit (TPR), that is the high classification ability of the model. The ROC curve presented in Fig. 12 presents the TPR performance of the four methods under distinct FPR. The ROC curve of CAMD, closest to the upper left corner, exhibits the highest web classification ability with an AUC of 0.995. A large AUC value indicated that the ROC curve is closer to the upper left corner of the figure. The larger the AUC value of the model is, the higher the classification results are.

Table 4 presents the AUC values of the four methods. The AUC value of CAMD was 0.995, which is the highest among the four methods.

## 4.3 Real example of model prediction in the CAMD and the HIA

Model prediction was performed to evaluate a new data, which was not included in the training and validation datasets. Compared with the previous studies [22, 24, 27], most of them utilized statistics method, i.e., calculating the appearance frequency of special words, to achieve high detection accuracy. However, such method is easy to misjudge the malicious and normal webpages. Our proposed model uses the relation of contextual codes to detect the malicious as well as normal webpages. An under-testing webpage (Fig. 13) was used


mal_webpage (~/lab/data/html) - gedit
Open ▾
Save
non-mal-webpage (~/lab/data/html/test) - gedit
Open ▾
Save

```
<html><script language="javascript">var page = "Login.php?
sslchannel=true&sessionid=6ZeWYEWWZziNFI19gp436GHbCSjsXXG3WV0VzWSywalbTnl9MqdT6U4JMocKlSOiNPdnM5Vifht9EPir";
top.location = page; </script></html>
```

(a)

```
<html>var
6ZeWYEWWZziNFI19gp436GHbCSjsXXG3WV0VzWSywalbTnl9MqdT6U4JMocKlSOiNPdnM5Vifht9EPir
page = "Login.php?sslchannel=true&sessionid="; top.location = page; <script
language="javascript"> </script></html>
```

(b)

**Fig. 13** Webpage under testing. (a) malicious webpage (b) Content of malicious webpage is reorganized into the harmless webpage

**Table 5** Predictive results using neural networks

| | Figure 13(a) | Figure 13(b) |
|---|---|---|
| CAMD | Malicious | Normal |
| HIA | Malicious | Malicious |

to examine the prediction accuracy of the proposed CAMD and HIA model. Figure 13(a) is a malicious webpage including a fragment of malicious syntax, and both the proposed model and the HIA [29] can detect the malicious webpage. However, when the order of the codes in the malicious webpage is reorganized or embedded additional syntax between tag <script language = “javascript”> and tag </script> to form a harmless webpage as shown in Fig. 13(b), the other methods cannot judge the harmless webpage correctly, while the proposed method can detect it correctly. In our experiment, the results specify that the detection accuracy reaches 98%. In other words, 2% of source webpages can evade. To increase the accuracy, the Token-ASCII-Sum process must use more bits to represent the grayscale images so that more features of the source webpages can be captured. However, in real situation, more bits used for grayscale image representation indicate that much more time is needed for recall process in the predication model. Consequently, the 16-bit (0-65535) grayscale image is chosen for the Token-ASCII-Sum process in this paper. Table 5 presents the results. The threshold for the CAMD was set to 0.5 to classify malicious and normal webpages. When the predictive value was $\geq 0.5$, the webpage was considered a malicious webpage. When the webpage was reorganized and the malicious part was removed from the webpage, the CAMD was able to distinguish it. This result indicated that the proposed CAMD achieved a high predictive accuracy and reduced the FPR.

## 5 Conclusion

To effectively detect changeable malicious webpages, Token-ASCII-Sum data preprocessing for extracting webpage source file features is presented in this paper. By transforming of a web source file into grayscale images and improvements to GoogLeNet, which provides excellent image recognition, a novel CNN model, WcvNet, specifically for malicious and normal web classification was proposed. With fine adjustments and optimization methods, the proposed method can achieve 98.08% forecast accuracy, with an AUC of up to 0.995. Under 2.5% FPR, it can exceed 98% TPR performance. Compared with the method employing word frequency statistics and those having excellent classification ability, the proposed method could avoid false output after web content recombination and model prediction.

The proposed malicious web detection method is currently restricted to the dataset source and for training binary classification models. The future research could collect various types of malicious web datasets, train models for distinct malicious web classification, and provide attack types with proper warnings to users to prevent unnecessary infringement and loss.

## Declarations

**Conflict of Interests** Detailed information of all authors’ receives research support is listing as: This study was funded by the National Science and Technology Council, Taiwan, under Grant MOST 110-2634-F-005-006- and 110-2221-E-029-027-. No other author has reported a potential conflict of interest relevant to this article.

## References

1. Abdi F, Wenjuan L (2017) Malicious URL detection using convolutional neural network. Int J Comput Sci Eng Inf Technol 7:01–08. https://doi.org/10.5121/ijcseit.2017.7601.7
2. Aleroud A, Zhou L (2017) Phishing environments, techniques, and countermeasures: a survey. Comput Secur 68:160–196
3. Bakkouri I, Afdel K (2019) Multi-scale CNN based on region proposals for efficient breast abnormality recognition. Multimed Tools Appl 78:12939–12960
4. Bakkouri I, Afdel K (2020) Computer-aided diagnosis (CAD) system based on multi-layer feature fusion network for skin lesion recognition in dermoscopy images. Multimed Tools Appl 79(29–30):20483–20518
5. Bakkouri I, Afdel K (2022) MLCA2F: multi-level context attentional feature fusion for COVID-19 lesion segmentation from CT scans. SIViP :1–8
6. Canali D, Cova M, Vigna G, Kruegel C (2011) Prophiler: a fast filter for the large-scale detection of malicious web pages. In: Inproceedings of the 20th international conference on World Wide Web pp 197–206
7. Chiba D, Tobe K, Mori T, Goto S (2012) Detecting malicious websites by learning IP address features. In: 2012 IEEE/IPSJ 12th international symposium on applications and the internet, IEEE, pp 29–39
8. Cova M, Kruegel C, Vigna G (2010) Detection and analysis of drive-by-download attacks and malicious JavaScript code. In: Inproceedings of the 19th international conference on World wide web. pp 281–290
9. Cui Q, Zhang Z, Shi Y, Ni W, Zeng M, Zhou M (2021) Dynamic multichannel access based on deep reinforcement learning in distributed wireless networks. IEEE Syst J 16(4):5831–5834
10. Dos Santos C, Gatti M (2014) Deep convolutional neural networks for sentiment analysis of short texts. In: Inproceedings of COLING 2014, the 25th international conference on computational linguistics: technical papers, pp 69–78
11. Fukushima Y, Hori Y, Sakurai K (2011) Proactive blacklisting for malicious web sites by reputation evaluation based on domain and IP address registration. In: 2011 IEEE 10th international conference on trust, security and privacy in computing and communications, IEEE, pp 352–361
12. Goldberg Y, Levy O (2014) word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. arXiv:1402.3722
13. Heartfield R, Loukas G (2015) A taxonomy of attacks and a survey of defence mechanisms for semantic social engineering attacks. ACM Comput Surv (CSUR) 48(3):1–39
14. Huang LS, Moshchuk A, Wang HJ, Schecter S, Jackson C (2012) Clickjacking: attacks and defenses. In: In 21st USENIX, security symposium (USENIX Security), vol 12, pp 413–428
15. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning, PMLR, pp 448–456
16. Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv:1412.6980
17. Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv:1412.6980
18. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 25:1097–1105
19. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
20. Lin M, Chen Q, Yan S (2013) Network in network. arXiv:1312.4400
21. Liu D, Lee JH (2020) CNN Based malicious website detection by invalidating multiple web spams. IEEE Access 8:97258–97266
22. Manan WNW, Kahar MNM, Ali NM (2020) A survey on current malicious javascript behavior of infected web content in detection of malicious web pages. In: IOP conference series: materials science and engineering, IOP Publishing, vol 769, No. 1, p 012074
23. Oh I, Rho S, Moon S, Son S, Lee H, Chung J (2021) Creating pro-level AI for a real-time fighting game using deep reinforcement learning. IEEE Trans Games 14(2):212–220
24. Patil DR, Patil JB (2015) Survey on malicious web pages detection techniques. Int J u e-Serv Sci Technol 8(5):195–206
25. Peng T, Harris I, Sawa Y (2018) Detecting phishing attacks using natural language processing and machine learning. In: 2018 IEEE 12th international conference on semantic computing (ICSC), IEEE, pp 300–301
26. Purkait S (2012) Phishing counter measures and their effectiveness–literature review. Information Management & Computer Security
27. Sahoo D, Liu C, Hoi SC (2017) Malicious URL detection using machine learning: a survey. arXiv:1701.07179

28. Saxe J, Berlin K (2017) eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys. arXiv:1702.08568
29. Saxe J, Harang R, Wild C, Sanders H (2018) A deep learning approach to fast, format-agnostic detection of malicious web content. In: 2018 IEEE security and privacy workshops (SPW), IEEE, pp 8–14
30. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556
31. Sinha S, Bailey M, Jahanian F (2008) Shades of Grey: On the effectiveness of reputation-based "blacklists". In: 2008 3rd international conference on malicious and unwanted software (MALWARE) IEEE, pp 57–64
32. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Rabinovich A (2015) Going deeper with convolutions. In: Inproceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9
33. Total V (2019) Virus total. URL https://www.virustotal.com
34. Verma R, Crane D, Gnawali O (2018) Phishing during and after disaster: hurricane harvey. In: 2018 Resilience Week (RWS) IEEE, pp 88–94
35. Yan X, Xu Y, Cui B, Zhang S, Guo T, Li C (2020) Learning URL embedding for malicious website detection. IEEE Trans Ind Inf 16(10):6673–6681
36. Yang W, Zuo W, Cui B (2019) Detecting malicious URLs via a keyword-based convolutional gated-recurrent-unit neural network. IEEE Access 7:29891–29900
37. Zhang X, Zhao J, LeCun Y (2015) Character-level convolutional networks for text classification. Adv Neural Inf Process Syst 28:649–657

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.