Computers & Security 136 (2024) 103561

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

PDHF: Effective phishing detection model combining optimal artificial and automatic deep features

Erzhou Zhu, Kang Cheng, Zhizheng Zhang, Huabin Wang *

School of Computer Science and Technology, Anhui University, Hefei 230601, PR China

ARTICLE INFO

Keywords:
Phishing detection
Phishing feature selection
Hybrid features
Convolutional neural network
Disorderly quantized attention mechanism

ABSTRACT

Currently, the increasing number of high-volume phishing attacks is among the largest threats to networking environments on a daily basis. During such a severe attack, researchers prefer to extract numerous features to improve the accuracy of phishing detection. However, the redundant features that may exist in the extracted feature set may be adapted by phishing attackers, not only degrading detection performance but also shortening the effective time of the constructed detection models. To address these problems, this study proposes phishing detection based on hybrid features (PDHF), a novel phishing detection model based on a combination of optimal artificial and automatic deep learning features. The optimal artificial phishing features are obtained by removing redundant features based on the newly designed feature importance evaluation index and an improved bidirectional search algorithm. To extend the effective time of phishing detection, deep features are learned from URLs using a one-dimensional character convolutional neural network (CNN) and a disorderly quantized attention mechanism. The experimental results show that PDHF outperforms many state-of-the-art methods and achieves an accuracy of 0.9965, precision of 0.9942, recall of 0.9940, and F_1-score of 0.9941. These results can help in the development of a security plug-in for clients, browsers, and various instant messaging tools that run on network edges, personal computers, smartphones, and other personal terminals.

1. Introduction

Phishing attempts lure recipients to provide sensitive information, by sending a large number of fraudulent links or spam claims from trusted institutions, such as banks, online retailers, and credit card companies (Mirian et al., 2019). According to the Anti-Phishing Working Group (APWG) (APWG, 2022) report for the fourth quarter of 2022, the number of phishing attacks is sharply increasing, as shown in Fig. 1. Phishing attacks are among the largest threats to networking environments (Jain and Gupta, 2022). During a severe threat, relying only on education is not sufficient to help users avoid phishing attacks; therefore, there is an urgent need for effective automatic detection methods to identify phishing websites (Chiew et al., 2018).

To alleviate the threat of phishing attacks, many feature-based automatic detection approaches have been proposed (Almomani et al., 2022). However, in these approaches, phishing detection performance relies not only on the number of features but also on the effectiveness of the extracted features (Zeng et al., 2020). In addition, phishing attackers can adapt many of these features over time (Liang et al., 2022). With the evolution of phishing, phishers are using new schemes to bypass detection, which necessitates the construction of a set of effective features. For example, feature f_7 listed in Table 3 (discussed in Section 6.2.1) is used to check if '@' exists in URLs. If a phishing detection system is built on feature f_7, the detected phishing website has a target URL containing '@'; however, if a phishing attacker can adapt to this feature and evade it, the constructed phishing detection system will lose its effectiveness. Thus, designing an automatic phishing feature deep learning model that can compensate for the defects of conventional phishing feature selection methods is important.

This study proposes phishing detection based on hybrid features (PDHF), a novel phishing detection model based on a combination of optimal artificial and automatic deep learning features. In the PDHF model, the optimal artificial feature set is first constructed sequentially using the information value (*InVa*) index and a simulated annealing (SA) based bidirectional search algorithm. Thereafter, effective deep-learning features are generated by the character convolutional neural network (CNN) and a disorderly quantized attention mechanism. Finally, optimal artificial and automatic deep features are combined to

* Corresponding author.
E-mail addresses: ezzhu@ahu.edu.cn (E. Zhu), ahu143813@163.com (K. Cheng), rousezz@163.com (Z. Zhang), wanghuabin@ahu.edu.cn (H. Wang).

https://doi.org/10.1016/j.cose.2023.103561
Received 22 October 2022; Received in revised form 19 October 2023; Accepted 21 October 2023
Available online 28 October 2023

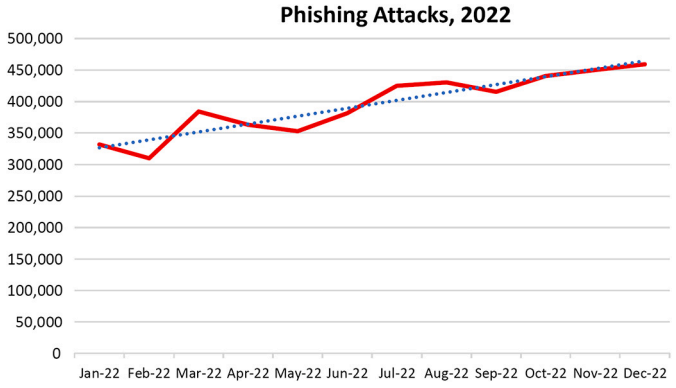

Fig. 1. APWG phishing activity trends report for the fourth quarter of 2022 (from APWG, 2022).

train the underlying random forest (RF) classifier for effective phishing detection. The contributions of this study can be summarized as follows:

(1) *Effective optimal artificial feature selection approaches*. In this study, the feature importance evaluation index *InVa*, which is based on the weight of evidence (*WoE*) feature encoding (the theory borrowed from the risk control of economics), is firstly designed to delete irrelevant features. Thereafter, the bidirectional search algorithm, which is revised by SA randomization, is designed to delete redundant features. Finally, the optimal artificial feature set is constructed by removing irrelevant and redundant features.
(2) *Automatic generation approaches for deep learning features*. In this study, we propose CNN-disorderly quantized attention (CNN-DQA), a deep learning model that is based on the one-dimensional character CNN and DQA mechanisms. CNN-DQA can automatically learn deep features from URL. To further improve the performance of phishing detection, the disorderly quantized attention mechanism is proposed to evaluate the importance and dependence of different URL parts simultaneously.
(3) *PDHF phishing detection model*. To extend the effective time of the machine learning method and further improve the performance of phishing detection, both optimal artificial and automatic deep features are incorporated into the PDHF phishing detection model. By combining the two types of features, a hybrid phishing feature set is constructed and used to train the underlying RF classifier.

The remainder of this paper is organized as follows. Section 2 presents related work. Section 3 provides an overview of the proposed PDHF phishing detection model. Section 4 presents the optimal artificial feature selection approach. Section 5 details the CNN-DQA deep learning model used in this research. In Section 6, the experimental results are presented and discussed. Finally, Section 7 concludes the paper and outlines directions for future research.

2. Related work

In addition to educating users for prompt identification of phishing activity (Lin et al., 2019), many software-based methods, such as blacklisting, visual similarity checking, heuristic-based detection, and data mining, have been proposed from different perspectives to alleviate the threat of phishing attacks (Khonji et al., 2013). To cope with the problem of phishing features that can be adapted by phishing attackers, deep learning models have been used to automatically learn phishing features (Chai et al., 2022).

2.1. Blacklist anti-phishing

A blacklist is created by collecting a considerable number of reported phishing URLs. An anti-phishing blacklist, ubiquitous in modern web browsers, is a crucial defense mechanism against large-scale phishing attacks. As a representative blacklist-based solution, Safe Browsing (Google, 2021) integrates the Google safe browsing application programming interface (APIs), to access the lists of unsafe web resources. Safe Browsing is now running in browsers such as Chrome, Firefox, and Safari. Similarly, PhishTank (OpenDNS, 2022) is maintained by OpenDNS and runs on the Opera cross-platform browser. SmartScreen (Microsoft, 2023) is Microsoft's solution for Internet Explorer and Edge browsers. Bell and Komisarczuk (2020) presented a measurement study on three key phishing blacklists: Google Safe Browsing, OpenPhish (2018), and PhishTank, to determine uptake, dropout, typical lifetimes, and overlap of URLs in these blacklists.

Several studies have been conducted to maintain the effectiveness of blacklists. Under certain circumstances, the substrings of malicious URLs can be partially mutated by phishing attackers to avoid blacklisting. Akiyama et al. (2011) proposed a heuristic method to improve blacklisting based on the assumption that unknown malicious URLs exist in the neighborhood of known malicious URLs created by the same adversary. Based on the observation that methodical long-term empirical measurement is an effective strategy for proactively detecting the weaknesses in anti-phishing ecosystems, Oest et al. (2020) proposed a PhishTime framework to continuously evaluate the effectiveness of anti-phishing blacklists. The PhishFarm (Oest et al., 2019) framework, proposed to measure the effectiveness of evasion techniques against browser-phishing blacklists, employs cloaking techniques to represent real attacks. In contrast, Azeez et al. (2021) proposed an automatically updated whitelist method for phishing detection by conducting a comparative analysis of visual and actual links.

Blacklisting methods do not need to access web content; therefore, they are very efficient in anti-phishing. However, updating the blacklist takes time, and the latest URLs may not be processed. The effectiveness of blacklists depends on their size, scope, update frequency, and accuracy, among other characteristics (Bell and Komisarczuk, 2020).

2.2. Phishing detection based on visual similarity checking

Based on the assumption that most phishing websites are similar or identical in visual appearance to legitimate websites, visual similarity checking is widely used for phishing detection. Bozkir and Aydos (2020) proposed a scheme for recognizing the brands of zero-hour phishing webpages by localizing and classifying the target brand logos involved in page screenshots using computer vision methods in object detection. In this scheme, the histogram features of oriented gradients are employed to obtain visual representations of the target brand logos in a scale-invariant manner. To improve phishing detection accuracy, Haruta et al. (2017) proposed a visual similarity-based phishing website detection scheme using images and cascading style sheet (CSS) with a target website finder, whereby websites were identified based on their similarities to the phishing websites. Abdelnabi et al. (2020) proposed VisualPhishNet, which is a similarity-based phishing detection framework based on a triplet CNN. VisualPhishNet learns website profiles to later detect these profiles using a similarity metric that can be generalized to pages with a new visual appearance. Visual similarity checking is accurate in phishing detection when the legitimate website corresponding to the phishing site is already registered in the built phishing detection system (Jai and Gupta, 2017), which incurs a large computational load on host platforms.

2.3. Heuristics-based phishing detection

Heuristic-based approaches have been developed to identify phishing through prediction, whereby a set of features from URLs, the layout of webpages, HTML codes, and JavaScript are extracted to compensate for the knowledge necessary to classify a website as malicious (Silva et al., 2020). Ding et al. (2019) first used a page title tag as a keyword for a webpage to quickly filter legal webpages using the Baidu search

engine. Seven heuristic rules based on the characteristics of phishing URLs were defined to determine whether a suspicious webpage violates these rules. Gupta and Jain (2020) presented a search engine-based phishing detection method which identifies phishing webpages accurately regardless of the textual language used within the webpage. In the search engine, a lightweight, consistent, and language independent search query is used to determine the legality of the suspicious URL. By calculating the heuristic value of the primary domain and that of the subdomain/pathdomain, Phishing-Aware (Pham et al., 2018) detects phishing attacks in fog networks. Comparing gain, loss, and combined gain and loss persuasion cues with a baseline model, Valecha et al. (2022) used the persuasion cues to detect phishing emails. Heuristic detection is a method of finding abnormal behaviors using practical experience. Heuristic rules for detecting phishing are designed based on typical abnormal behaviors and statistical analysis results. Compared to blacklist technology, these rules can cope with newly emerging phishing URLs. However, phishing attackers can bypass the constructed filters after obtaining the heuristic rules.

2.4. Phishing detection based on data mining

In data mining-based phishing detection models, the detection process is generally divided into two steps: feature extraction and categorization. First, various representative features are extracted to capture the website characteristics. Second, intelligent techniques are used to automatically categorize websites into different classes based on a computational analysis of feature representations (Zhuang et al., 2012). Wu et al. (2022) proposed an approach for detecting phishing scams on Ethereum by mining large-scale transaction records. This approach identifies phishing nodes using features extracted from the Ethereum transaction history and newly designed trans2vec network embedding algorithms. Data mining approaches consider the detection of phishing attacks as a document classification or clustering problem, and the data models are constructed based on machine learning and clustering algorithms (Khonji et al., 2013). However, clustering algorithms, such as K-means and K-nearest neighbors (KNN), partition the target datasets in an unsupervised manner, improperly setting the number of clusters and initial clustering centers, which can severely degrade the quality of clustering results.

2.5. Phishing detection based on selecting optimal features

For feature-based detection approaches, the phishing detection performance relies not only on the number of features but also on the effectiveness of the extracted features. For many machine learning-based classification problems, features have a major influence on accuracy and time efficiency (Mishra et al., 2021). Zeng et al. (2020) developed PhishBench, a framework that allows researchers to rapidly evaluate newly collected features and design methods for machine learning-based phishing detection. Chiew et al. (2019) proposed a two-stage feature-selection framework for machine learning-based phishing detection systems. The first stage uses the cumulative distribution function gradient (CDF-g) algorithm to produce primary feature subsets. In the second phase, a set of baseline features is derived from secondary feature subsets using a function-perturbation ensemble. By extending the Gini coefficient and utilizing a decision tree, a feature evaluation index is defined to select the optimal features for a decision tree optimal features artificial neural network (DTOF/ANN) (Zhu et al., 2020) phishing detection model. Based on a Gaussian kernel, Zouina and Outtaj (2017) defined a “similarity” index for evaluating the features extracted from URLs. PHISH-SAFE (Jain and Gupta, 2015) selects 14 important features from URLs for accurate phishing detection. Zabihimayvan and Doran (2019) applied the fuzzy rough set theory to select the most effective features from three benchmark phishing datasets.

Different feature selection methods or evaluation indicators generate different optimal features. Four classes of features related to URL, domain, HTML/Javascript, and abnormal activities (Mohammad et al., 2014), are commonly extracted by many of the existing phishing detection models. From the address bar (URL), common features such as the length in character, port, and the protocol used to transfer information; from the domain, related features such as the age of registered domain names, the traffic of websites, and the number of links pointing to target webpages; from the HTML/Javascript code, features like the number of webpage and Iframe redirections; and for features of abnormal activities on a webpage, loading embedded objects from external domains or submissions to an email address are commonly extracted.

2.6. Deep learning-based phishing detection

Deep-learning methods such as CNN, recurrent neural network (RNN), and long short-term memory (LSTM) can automatically extract features from input data. Currently, deep learning methods are widely used for phishing detection. Liang et al. (2022) proposed CyberLen, a deep-learning-based system for detecting malicious URLs. In the CyberLen system, a factorization machine (FM) is used to learn latent interactions among lexical features. The phishing detection method proposed by Li et al. (2022) comprises two stages: sample expansion and testing with sufficient samples. In the sample expansion stage, the K-nearest neighbors method is combined with K-means to expand the training dataset such that the size of the training samples satisfies the requirements of in-depth learning. During the testing stage, the samples are preprocessed through generalization, word segmentation, and word vector generation. Based on software-defined network (SDN) technology, a clustering and feature method, and a CNN algorithm, Wazirali et al. (2021) proposed a URL phishing detection method that does not require retrieving the content of a target website or using any third-party services. In this method, recursive feature elimination (RFE) with a support vector machine (SVM) algorithm is used to select phishing features. Based on the analysis of text content and URL in the short message service (SMS), Jain et al. (2022) integrated KNN and RF to detect smishing SMS in intelligent mobile systems.

To provide more resources on important deep features, the attention mechanism is widely utilized in deep phishing detection models. Xiao et al. (2020) introduced multi-head self-attention into a one-dimensional CNN model for high-performance phishing website detection. Chai et al. (2022) proposed a multi-modal hierarchical attention model that jointly learns deep fraud cues from the content of webpages, navigational content of URLs, and visual content of images for phishing website detection. He et al. (2021) proposed DatingSec, a system for detecting malicious accounts in dating applications. This system leverages LSTM and an attention module to capture the interplay between users’ temporal-spatial behaviors and user-generated textual content.

2.7. Discussion

Owing to the powerful learning and classification abilities of machine learning approaches, machine learning is widely used in phishing detection models (as in many of the works discussed in Section 2.2-2.4) that consider phishing detection as a classification problem (Jain and Gupta, 2016). Machine learning-based anti-phishing approaches are generally based on feature engineering. Researchers prefer to extract a considerable number of features to improve the accuracy of phishing detection. For example, Off-the-hook (Marchal et al., 2017) approach extracted more than 200 features to accurately detect phishing attacks. However, an excessive number of irrelevant and redundant features in the extracted feature set can not only consume a large number of computing resources but also easily cause overfitting of the underlying classifiers. To construct the optimal artificial feature set, as discussed in Section 2.5, in this study, we sequentially used the new proposed *InVa* index and a simulated annealing revised bidirectional search algorithm.

Phishing attackers can adapt many of these features over time (Liang et al., 2022). Thus, it is difficult to maintain the effectiveness of artificial

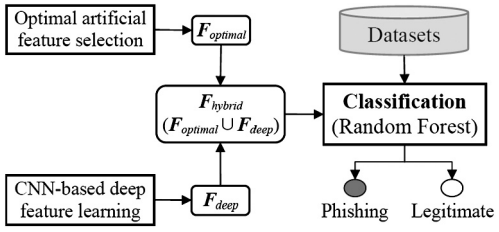

Fig. 2. Architecture of the PDHF model.

feature-based phishing detection models over long periods. As discussed in Section 2.6, deep features are automatically learned according to URL strings, HTML code, and other components of the website. Phishing detection models built on deep features are effective much longer than those based on artificial feature-based phishing detection systems, as the former cannot easily be adapted by attackers. The deep learning method has low computational overhead and high accuracy. However, the quality of automatic deep learning features depends highly on the scale of the datasets to be learned.

Therefore, it is important to design an automatic phishing feature deep learning model that can compensate for the defect (short validity period caused by extracted features that can be easily adapted by attackers) in artificial phishing feature selection methods. Unlike many of the existing methods, in this study, both optimal artificial and automatic deep features are incorporated into the PDHF phishing detection model. By combining the two kinds of features, the high phishing detection accuracy of the PDHF can be maintained for longer period than the artificial feature-based phishing detection methods. The training dataset for the CNN-DQA is smaller than those of the existing deep learning phishing detection methods, as it is only used as an aid in determining optimal artificial features, to increase the difficulty for phishing attackers to adapt hybrid features.

3. Overview of the PDHF phishing detection model

Fig. 2 shows the architecture of the PDHF phishing detection model. In this figure, the "optimal artificial feature selection" module (discussed in Section 4) specifies the artificial optimal feature selection method. In feature-based phishing detection systems, several features are collected to characterize phishing attacks. However, artificially collected features contain many irrelevant and redundant features. To eliminate the influence of irrelevant and redundant features, we propose the *InVa* feature evaluation index and an SA-based bidirectional feature selection algorithm. Using these two mechanisms, the optimal selected features ($\boldsymbol{F}_{optimal}$, with bold characters representing vectors or matrices in this study) reduce the burden on the underlying classifiers without adversely affecting phishing detection accuracy.

The "CNN-based deep feature learning" module (discussed in Section 5) specifies the deep feature learning approaches to use, to automatically extract phishing features. In this study, the proposed CNN-DQA automatically learns deep features ($\boldsymbol{F}_{deep}$) from target URLs.

To extend the timeliness of the machine learning method and further improve the performance of phishing detection, both optimal artificial and automatic deep features were incorporated into the PDHF phishing detection model. As shown in Equation (1), by combining the optimal artificial features ($\boldsymbol{F}_{optimal}$) and the automatic deep features ($\boldsymbol{F}_{deep}$), hybrid features ($\boldsymbol{F}_{hybrid}$) are generated. Then, $\boldsymbol{F}_{hybrid}$ is used to train the underlying RF classifier. Finally, the trained classifier is used to detect phishing attacks.

$$\boldsymbol{F}_{hybrid} = \boldsymbol{F}_{optimal} \cup \boldsymbol{F}_{deep} \tag{1}$$

4. Optimal artificial phishing feature selection

This section presents approaches for selecting optimal features from an artificially collected feature set (the phishing features related to this study are discussed in Section 6.2.1). As shown in Fig. 3, the optimal bin segmentation algorithm, which is based on the extended Gini coefficient and the CART algorithm, is first designed to discretize the continuous phishing features. Thereafter, the *InVa* feature importance evaluation index based on *WoE* feature encoding is designed to delete irrelevant features. Third, a bidirectional search algorithm, which was revised using simulated annealing (SA) randomization, is designed to delete redundant features. Finally, the optimal feature set is constructed by removing irrelevant and redundant features.

4.1. Continuous feature discretization

Continuous feature discretization In practice, different parts of continuous features play various roles in improving the accuracy of phishing detection. It is necessary to discretize the continuous phishing features. Discretization is the segmentation of continuous features into several segments (also known as bins). In this study, the Gini coefficient was extended to process the phishing features; the principle of generating feature segmentation points originally used in the construction of CART was applied to optimally divide continuous phishing features into several bins.

In the fields of economics and sociology, the Gini coefficient is used to measure the degree of inequality in income or wealth distribution. The purpose of introducing the Gini coefficient into the decision-tree algorithm is to measure the purity of the dataset, that is, the degree to which the samples in the dataset belong to the same category. The smaller the Gini coefficient, the higher is the purity of the dataset, and thus, the better is the classification performance of the decision tree algorithm on the dataset. This study used the Gini coefficient to segment continuous features and explored the importance of the features at different intervals. Specifically, the Gini coefficient of the dataset D can be defined as

$$Gini(D) = 1 - \sum_{k=1}^{K} p_k^2, \tag{2}$$

where K is the number of classes in the current dataset; D should be divided, and p_k is the proportion of samples of the k^{th} class among the total samples of D. In phishing detection, the samples in dataset D can be divided into two classes: phishing and legitimate. Therefore, the value of K in Equation (2) is fixed at two. The Gini coefficient reflects the probability of randomly taking two samples from dataset D and marking them as not belonging to a category. The smaller the value of the *Gini(D)*, the higher is the purity of the dataset D.

Suppose that f is a feature of phishing dataset D, and it has V possible values $A = \{a^1, a^2, ..., a^V\}$. According to the CART algorithm (Sundhari, 2011), the decision tree has V branch nodes when feature f is used to divide dataset D. The v^{th} $(v = 1, 2, ..., V)$ branch node accommodates all samples with a^v values on feature f. The number of samples in the node is denoted by D^v. As samples can only be divided into two classes, the phishing dataset D is divided into D_1 and D_2 when a^v is selected as the feature segmentation point. In phishing dataset D, the Gini coefficient of f when a^v is taken as the segmentation point is defined as

$$Gini(D, a^v) = (|D_1|Gini(|D_1|) + |D_2|Gini(|D_2|))/|D|, \tag{3}$$

where $|D_1|$, $|D_2|$, and $|D|$ denote the number of samples in the datasets D_1, D_2, and D, respectively.

We can calculate the Gini coefficients of f for all possible values using Equation (3). The optimal segmentation point with the smallest Gini coefficient is defined as

$$f\_a_*^v = min_{a^v \in A} Gini(D, a^v),\ v = 1, 2, ..., V. \tag{4}$$

For each set composed of samples in the corresponding branch node, the same method was used to divide the set into several smaller sets until the number of samples in each set was smaller than the minimum

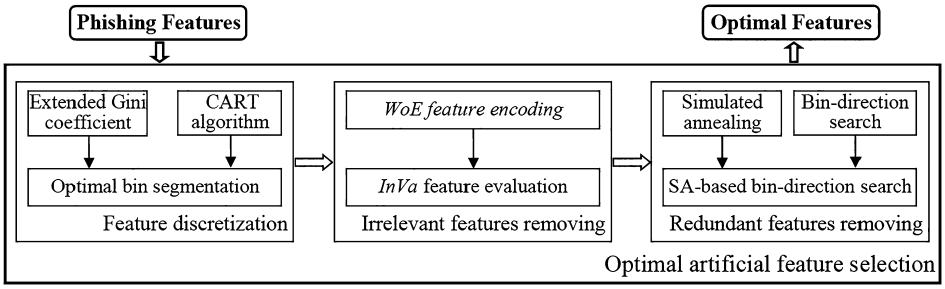

Fig. 3. Workflow of selecting optimal features from the artificial collected feature set.

Algorithm 1: Optimal feature bin segmentation based on the revised Gini coefficient.

Input: Phishing dataset $D=\{(x_1,y_1),(x_2,y_2),...,(x_n,y_n)\}$, Feature set $D=\{f_1,f_2,...,f_m\}$.
Output: Segmentation bins $=B=\{b_1,b_2,...,b_d\}$ of each feature $f_i, i=1,2,...,m$.
1: **for** each feature $f_i \in F$ **do**
2: Following an ascending order, the V possible values of feature f are ordered as $\{a^1,a^2,...,a^V\}$;
3: By calculating the average value of two adjacent values of f and taking it as the segmentation point, the V-1 segmentation points are generated as $\{b^1,b^2,...,b^{V-1}\}$;
4: Based on Equation (3), the Gini coefficient of f is obtained when the value $b^v(v=1,2,...,V-1)$ is taken as the segmentation point;
5: Based on Equation (4), the optimal segmentation point with the smallest Gini coefficient is used to divide the dataset D into D_1 and D_2;
6: Based on Equation (5), the minimum sample size of the subset, $|D_v|_{min}$, is calculated;
7: if $|D_1|>=|D^v|_{min}$ recursive partitioning dataset D_1;
8: if $|D_2|>=|D^v|_{min}$ recursive partitioning dataset D_2;
9: **end for**

sample size. In this study, the minimum sample size of the subsets was defined as

$$|D^v|_{min}=|D|\times\rho, \tag{5}$$

where ρ is the D-segmentation index. As a rule of thumb, ρ is typically set to 0.1.

Based on the above analysis, an algorithm for optimal bin segmentation based on the revised Gini coefficient was designed, as shown in Algorithm 1. In this algorithm, the original phishing dataset $D=\{(x_1,y_1),(x_2,y_2),...,(x_n,y_n)\}$ contains the n phishing samples, and each sample has m features. Moreover, label y_i specifies the classification result (legal or illegal) for the sample x_i. The datasets used in this study are described in Section 6.1. Line 2 ascends the V possible values of feature f. Line 3 calculates the average of two adjacent values of feature f and considers it as the segmentation point. According to Equation (3), line 4 calculates the Gini coefficient of f when b^v is considered the segmentation point. Based on Equation (4), line 5 determines the optimal segmentation point with the smallest Gini coefficient and uses it to divide dataset D into D_1 and D_2. Based on Equation (5), line 6 calculates the minimum sample size of the subset ($|D^v|_{min}$). Lines 7 and 8 partition the generated subsets recursively until their sample sizes are smaller than $|D^v|_{min}$.

4.2. Optimal feature selection

In this study, a two-stage feature selection method was proposed to delete irrelevant and redundant features. In the first stage, the *InVa* feature importance evaluation index, which is based on the *WoE* feature encoding approach, was used to delete irrelevant features. In the second stage, an SA-based bidirectional search approach was proposed to delete redundant features. Subsequently, an optimal set of phishing features was constructed.

4.2.1. Irrelevant feature elimination

WoE is used to calculate the correlation between independent and dependent variables. This measures the amount of information in the attributes of the independent variable. Each feature was assigned a value based on *WoE*. This value is then considered the degree to which the classification results are differentiated. In this study, using discretization operations, each continuous feature f is segmented into several bins $B=\{b_1,b_2,...,b_d\}$. For each bin b_i of feature f, the value of *WoE* is computed and used to quantify the difference between the observed proportion of legitimate and phishing websites in this box. Specifically, suppose that the phishing feature f is segmented into d bins $B=\{b_1,b_2,...,b_d\}$; P_i and L_i specify the number of phishing and legitimate samples in the i^{th} bin; the value of *WoE* under bin b_i can then be calculated as

$$WoE(f,i)=ln(P_i/\sum_{i=1}^{d}P_i)-ln(L_i/\sum_{i=1}^{d}L_i). \tag{6}$$

The farther the value of *WoE(f, i)* is from zero, the more information bin b_i contains. Because the values of the bins of f are computed from *WoE*, the missing features and outliers of the features in the original phishing datasets can be eliminated.

Based on the *WoE* encoding, *InVa* was designed to evaluate the importance of phishing features. According to the *WoE* of the d bins, the *InVa* of feature f can be calculated as

$$InVa(f)=\sum_{i=1}^{d}(P_i/\sum_{i=1}^{d}P_i-L_i/\sum_{i=1}^{d}L_i)\times WoE(f,i). \tag{7}$$

According to the rule of thumb (Kuo et al., 2022) in the risk control of economics, the value of *InVa(f)* can be expressed as

- $InVa(f)<0.05$: Feature f has little prediction ability;
- $0.05\le InVa(f)<0.1$: Feature f has low prediction ability;
- $0.1\le InVa(f)<0.25$: Feature f has medium prediction ability;
- $InVa(f)\ge 0.25$: Feature f has strong prediction ability.

In this study, according to *InVa* values, the phishing features were first ordered. All phishing features with *InVa* values smaller than 0.1 were considered irrelevant. As irrelevant features have almost zero prediction ability, they were deleted to complete the first stage of phishing feature selection.

4.2.2. Removal of redundant features and construction of optimal feature vector

Using a single feature evaluation index, useless features can be effectively filtered out. However, there may be correlations between features, and strongly correlated features are considered redundant. Redundant features have a negative effect on classification and increase the complexity of the underlying classifiers. In this section, we propose a bidirectional search approach that integrates a simulated annealing randomization strategy to eliminate redundant features.

A bidirectional feature search is a combination of forward and backward feature search approaches. In the initial stage of forward feature search strategy, the target feature set S is empty. The candidate feature set X accommodates all features that must be simplified. According to an evaluation criterion, the forward feature search strategy selects the current optimal feature from X and puts it into S stepwise until all optimal features are placed into S. In contrast, in the initial stage of the backward feature search strategy, the target feature set S is also a candidate feature set that accommodates all features to be simplified, gradually deleting one feature at a time under the evaluation criterion until all unimportant features are removed from S.

In the bidirectional feature search approach, the order of the optimal feature generation is bidirectional. In this approach, the final optimal feature set is generated by adding important features from the candidate set X or removing unimportant features from the target set S. The bidirectional feature search approach incorporates two stages. The first stage is adapted to a forward search strategy. At this stage, the features in candidate set X are selected. If the value of the evaluation function increases after adding feature X, this feature is placed in target set S. The second stage is adapted to a backward search strategy. At this stage, the features in target set S are selected. If the value of the evaluation function (which may differ from the evaluation function used in the first stage) is not clearly reduced after removing a feature from S, then the feature is moved from the target set S to the candidate set X.

The bidirectional feature-search approach has the advantages of easy implementation and relatively low computational complexity. However, it is easy to fall into the problem of local optima. To solve this problem, the concept of simulated annealing (Kuo et al., 2022), which is a random search strategy, was introduced to revise the approach.

In each iteration of the first forward stage of the revised bidirectional feature search, feature f is randomly selected from the candidate set X. The nonoptimal solution at the current temperature (i.e., the solution involving the new feature f) is accepted by probability, whereby the problem of local optimality in conventional sequence searches is avoided. In each iteration of the second backward stage of the revised bidirectional feature search, feature f is first randomly selected from the target set S. Thereafter, the values of the evaluation function before and after removing feature f from S are calculated. Finally, if the value of the evaluation function after removing f from S is larger than the value of this function in the original S, feature f is moved from the target set S to the candidate set X. Otherwise, the selected feature f is retained in target set S.

Based on the above analyses, a bidirectional search approach incorporating a simulated annealing randomization strategy is proposed to eliminate redundant features. Algorithm 2 illustrates the main steps of this approach. Using simulated annealing revision, the features in the two stages of the bidirectional search are randomly selected. New solutions are also randomly generated for the target set S. In the first feature-addition stage, a simulated annealing strategy is introduced to ensure the probability of breaking through local optimization. This fully considers the correlations among the different features. The second stage ensures that redundant features are deleted.

5. Generation of automatic deep learning phishing features

In this section, CNN-DQA, which is a deep learning model based on a one-dimensional character CNN and a disorderly quantized attention mechanism, is designed to automatically learn features from URLs. As shown in Fig. 4, the CNN-DQA model comprises a data preprocessor, as well as convolutional, disorderly quantized attention, maximum pooling, and full connection layers. This model can individually generate classification results for the target URLs. Moreover, the extracted deep features can be combined with previously selected optimal artificial features for many types of classifiers to achieve high phishing detection accuracy.

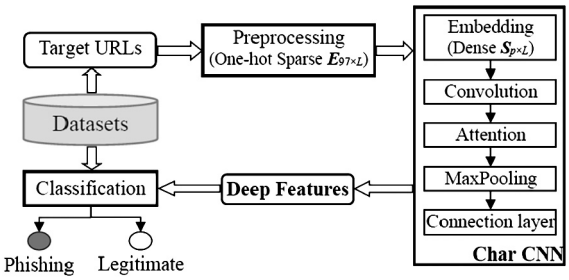

Fig. 4. Architecture of the CNN-DQA model.

5.1. URL preprocessing

During deep learning, URLs are used as strings. These strings are then divided into characters. Each split single character is finally converted into a vector. In this study, the one-hot encoding is used to vectorize the URL characters. In natural language processing, one-hot encoding is a common method for text (string) vectorization. The one-hot fragment representation of characters is similar to bitmap representation that uses 0/1 to represent the position where the character appears. Based on a statistical analysis of URLs in phishing datasets (the datasets used in this study are described in Section 6.1), as listed in Table 1, 95 frequently used characters, including 52 uppercase and lowercase letters, 10 numbers, and 33 special characters, were collected in this study. The characteristics of URLs were mapped to integers. The "*unk*" and "*pad*" are common identifiers in natural language processing studies, representing "unknown character" and "pad character," respectively. In this study, if a character appearing in the URLs was not among the collected 95 characters, it was set to "*unk*" (unknown). Because the neural network can only handle fixed-length vectors, the length of URLs was fixed to L with all excess characters deleted and vacant spots filled with "*pad*" characters.

According to statistics, as shown in Table 1, URLs typically consist of 95 common characters. Therefore, with the addition of unknown characters "*unk*" and padding characters "*pad*" required for unified and standardized processing of strings, each character was converted into a one-hot fragment $\boldsymbol{e}$ (In this article, bold variables represent the names of vectors or matrices.) with a fixed length of 97, with the corresponding position for this character set to 1, and other positions set to 0 in the one-hot fragment. In this study, a one-hot fragment is represented as a one-dimensional column vector. For instance, the code of '*a*' is 1, and it can be represented as $\boldsymbol{e} = (0, 1, 0, ..., 0)^T$. Accordingly, the integral URL U can be converted into a matrix $\boldsymbol{E}_{97\times L}$:

$$\boldsymbol{E} = (\boldsymbol{e}_1, \boldsymbol{e}_2, ..., \boldsymbol{e}_L). \tag{8}$$

For instance, the one-hot matrix of URL "*www.ahu.edu.cn*" is illustrated in Fig. 5. In this figure, each column represents a one-hot vector corresponding to URL characteristics.

5.2. Feature automatic learning

The one-hot encoded matrix $\boldsymbol{E}$ is high-dimensional and sparse with many zeros. To overcome this weakness, an embedding layer that projects each one-hot fragment $\boldsymbol{e}_i$ onto a low-dimensional dense character embedding space is introduced. The embedding layer can be

Algorithm 2: SA-based bidirectional feature search algorithm.

Input: Phishing dataset D, Candidate feature set X.
Output: Target optimal feature set S.
/* X is composed of features of D that are filtered by *InVa*. */
/* $F(S)$ is the evaluation function used as the parameter in the bidirectional search. In this study, $F(S) = accuracy\ (S)$. */
/* Simulated annealing parameters: T_0 is the initial temperature; T_F is the termination temperature; α is the cooling coefficient; N is the number of iterations for a temperature. */
$S \leftarrow \Phi; T \leftarrow T_0$; /* Initially, the target set S is empty; the current temperature T is set to T_0. */
while $T \neq T_F$ **do**
  for each iteration (total N iterations) of the current temperature T **do**
    Add phishing features in the forward direction:
    $f \leftarrow Random(X); X \leftarrow X - f$; /* Randomly select a feature f from X. */
    $S' \leftarrow S \cup \{f\}$; /* Put feature f into target set S.*/
    $\triangle T \leftarrow F(S') - F(S)$; /*Calculate the increment of evaluation function. */
    if $\triangle T > 0$ **then**
      $S \leftarrow S'$; /* S' is accepted as the new solution (the current optimal feature set). */
    else
      $S \leftarrow S'\_exp(-\triangle T/T)$; /* Accept S' as the new current solution with probability of $exp(-\triangle T/T)$. */
    end if
    Remove phishing features in the backward direction:
    $f \leftarrow Random(S)$; /* Randomly select a feature f from S. */
    $S' \leftarrow S - \{f\}$;
    $\triangle T \leftarrow F(S') - F(S)$; /* Calculate the decrement of evaluation function. */
    if $\triangle T < 0$ **then**
      $S \leftarrow S$; /* The current target solution S is not changed. */
    else
      $S \leftarrow S'$; /* Accept S' as the new current solution. */
    end if
    $T \leftarrow T \times \alpha$; /* Cooling. */
  end for
end while

Characters of a URL

one-hot fragment of a character	w	w	w	.	a	h	u	.	e	d	u	.	c	n	pad	…	pad
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14		*L*-1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0		0
…																	
3	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0		0
4	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0		0
5	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0		0
…																	
8	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0		0
…																	
14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0		0
…																	
21	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0		0
…																	
23	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0		0
…																	
90	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0		0
…																	
96	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0

Fig. 5. The one-hot bit matrix of a URL.

Table 1
Character-integer mapping.

Characters	Encodes
a b c d e f g h i j k l m n o p q r s t u v w x y z	1-26
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z	27-52
0 1 2 3 4 5 6 7 8 9	53-62
` ~ ! @ # $ % ^ & * ( ) - _ + = []{ } \ | ; : ‘ “ , . / < > ? &apos	63-95
unk	96
pad	0

considered as a dictionary with m indices that map integer indices (representing specific characters) onto p-dimensional dense vectors. The weight of the embedding layer is randomized at the beginning and adjusted using backward propagation during training.

Let $\boldsymbol{W}_{p\times 97}$ be the weight matrix. The embedding layer utilizes the following operations to convert each one-hot fragment $\boldsymbol{e}_i = (e_{i,1}, e_{i,2}, ..., e_{i,97})$ into a vector $\boldsymbol{s}_i = (s_{1,i}, s_{2,i}, ..., s_{p,i})$ with p elements $(i = 1, 2, ..., L)$:

$$\boldsymbol{S}_{p\times L} = \boldsymbol{W}_{p\times 97} \times \boldsymbol{E}_{97\times L}. \tag{9}$$

Each URL U, represented by the sparse one-hot fragment $\boldsymbol{E}_{97\times L}$, is converted into a dense matrix $\boldsymbol{S}_{p\times L}$. The parameters L and p are set through repeated experiments (Section 6.3). The value of the parameter p is smaller than 97, which implies that a reduction in the dimension p of the embedding layer can improve efficiency without data loss.

A single convolution kernel extracts only a local one-dimensional feature sequence from the feature matrix $\boldsymbol{S}_{p\times L}$ of the embedding layer. To obtain different scales of the local features, the sizes of the convolution kernels used in the convolution layer were set to q_1 or q_2. The number of different convolution kernels was set to 2^K to improve detection accuracy. In general, the larger the value of K, the richer is the extracted feature information. However, excessive convolution kernels impose a large computational burden on underlying platforms. In the

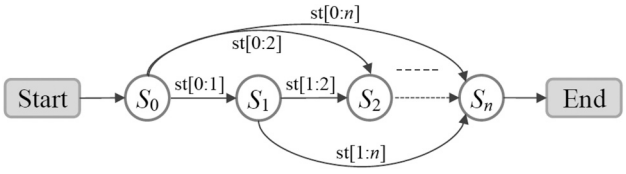

Fig. 6. Words extracted from string *str*.

experimental section (Section 6.3), the number of convolution kernels is set through repeated experiments.

After the convolution layer, the maximum pooling layer establishes the position invariance in a larger local region, in addition to sampling the input objects. In this process, the most important features are selected to improve the convolution performance. Convolution operations are applied to the maximum pooling layer to obtain output vectors of different scales.

$$\boldsymbol{y}_c^{q_1}(U) = Maxpool(Relu(Conv(\boldsymbol{w}_{cnn}^{q_1}, \boldsymbol{E}) + \boldsymbol{b}_{cnn}^{q_1})), \tag{10}$$

$$\boldsymbol{y}_c^{q_2}(U) = Maxpool(Relu(Conv(\boldsymbol{w}_{cnn}^{q_2}, E) + \boldsymbol{b}_{cnn}^{q_2})). \tag{11}$$

Finally, in the fully connected layer, the entire feature representation of URL U is generated by concatenating different types of previously extracted feature vectors:

$$\boldsymbol{y}_c(U) = Concat(\boldsymbol{y}_c^{q_1}(U), \boldsymbol{y}_c^{q_2}(U). \tag{12}$$

5.3. Disorderly quantized attention mechanism

In this section, a quantitative method is proposed for determining disorder in each part of the URL. The quantitative value of the disorder of each URL is then taken as the weight of the attention mechanism for constructing the entire feature vector. The disorderly quantized attention mechanism used in the deep-learning model can be described as follows.

5.3.1. Word extraction from URL

Let the target URL U be the string *str* composed of n characters, $st[i:j]$ be a substring composed of the i^{th} character to the j^{th} character of *str*, and $0 \le i < j \le n$. A string *str* is typically composed of several words with independent meanings. As shown in Fig. 6, in string *str*, if the positions before and after each character are considered a state, each path from the initial state S_0 to the termination state S_n forms a partition of *str*. Each edge of this path represents a word contained in *str*. Using this method, the words contained in the target URL U can be extracted as

$$\boldsymbol{WordU} = (u_1, u_2, ..., u_m). \tag{13}$$

5.3.2. Word cost assignment

In the corpora for natural language processing, according to Zipf's law (Piantadosi, 2014), the frequency of a word is inversely proportional to its ranking in the frequency table. Thus, based on frequency, the words in the corpus were ordered first. Then, the cost of a word, w_i, in the corpus can be defined as

$$cost(w_i) = ln(rank(w_i)) + 1, \tag{14}$$

where $rank(w_i)$ is the ranking of the number of occurrences of w_i in a corpus.

Based on the cost calculation of each word in the corpus for any given string, the minimum cost value *mc* can be calculated using the dynamic programming method as follows:

$$mc(st[i:j]) = \begin{cases} 0, i = j \\ cost(st[i:j]), cost(st[i:j]) \neq 0 \\ min_k\{mc(st[i:k]), mc(st[k:j])\}, else \end{cases}. \tag{15}$$

5.3.3. Attention weight of each URL part

According to Section 5.3.1, the m words contained in the target URL U are first extracted. Subsequently, the attention weight of each extracted word $u_i \in \boldsymbol{WordU}$ can be assigned as

$$Att(u_i) = mc(u_i). \tag{16}$$

The attention weight of the entire $\boldsymbol{WordU}$ can be expressed as

$$\boldsymbol{Att}(\boldsymbol{WordU}) = (Att(u_1), Att(u_2), ..., Att(u_m)). \tag{17}$$

5.3.4. Normalization of URL weighted vector

A sigmoid function is used to normalize the generated weighted vector $\boldsymbol{Att}(\boldsymbol{WordU})$ of the target URL U. As shown in Equation (18), the weighted value of each weighted URL part $Att(u_i)$ in Equation (17) is first multiplied by the constant $\lambda (0 < \lambda < 1)$. Subsequently, the normalized value of $Att\_N(u_i)$ is generated using a sigmoid function.

$$Att\_N(u_i) = sigmoid(\lambda \times Att(u_i)). \tag{18}$$

The normalized weighted vector $\boldsymbol{Att}(\boldsymbol{WordU})$ of target URL U is expressed as follows:

$$\boldsymbol{Att\_N}(\boldsymbol{WordU}) = (Att\_N(u_1), Att\_N(u_2), ..., Att\_N(u_m)). \tag{19}$$

5.3.5. Final vector generation

By performing the inner product operation on $\boldsymbol{Att\_N}(\boldsymbol{WordU})$ (Equation (19)) and $\boldsymbol{y}_c(U)$ (Equation (12)), the final vector representation of URL U can be obtained by

$$\boldsymbol{F}_{deep}(U) = \boldsymbol{Att\_N}(\boldsymbol{WordU}) \cdot \boldsymbol{y}_c(U). \tag{20}$$

5.4. CNN classification

The linear layer and sigmoid activation function determine the classification result (legitimate or phishing) of the final representation $\boldsymbol{F}_{deep}(U)$ for the target URL U. In Equation (21), *logit* indicates the probability that target URL U belongs to a phishing website.

$$logit = sigmoid(\boldsymbol{w}_{logit}) \times \boldsymbol{F}_{deep}(U) + \boldsymbol{b}_{logit}. \tag{21}$$

Here, feature vector $\boldsymbol{F}_{deep}(U)$ is redirected to the neural network, where each neuron generates a number. These numbers are processed using the sigmoid activation function to generate a probability value *logit*. The value of *logit* is the probability of specifying the features of the target URL U. If the value of *logit* is greater than 0.5, then the target URL U is a phishing URL. Otherwise, target URL U is legitimate.

6. Experimental results

The PDHF model is based on optimal artificial feature selection and automatic deep feature generation. Therefore, the effectiveness of these two mechanisms was evaluated first. Subsequently, the performance of the entire model was evaluated and compared with those of existing deep- or machine-learning-based phishing detection methods. The experiments were performed on a computer with an Intel i7 CPU, NVIDIA GeForce RTX 3080 GPU, DDR3 16 GB RAM, and Windows 11 operation system. The experimental programs were written using the Python programming language.

6.1. Datasets and metrics for the performance evaluation

As listed in Table 2, two datasets with different scales were used to evaluate PDHF. The first dataset, DATA1, was collected by Tan (Tan, 2018) and contained 10,000 samples among which 5000 were phishing, and 5000 were legitimate. This dataset has multiple sources, including PhishTank, OpenPhish, Alexa, and Common Crawl.

The second dataset, DATA2, is an assembled dataset (on October 20, 2021) that contains 1,172,577 URLs with 587,668 phishing and

Table 2
Description of Datasets.

Datasets	Categories	Samples	Sources	Access
DATA1	Phishing Legitimate	5000 5000	(Tan, 2018)	Directly downloaded
DATA2	Phishing Legitimate	587668 584909	PhishTank, Reasonable Antiphishing Alexa, CNSTC	Synthesized

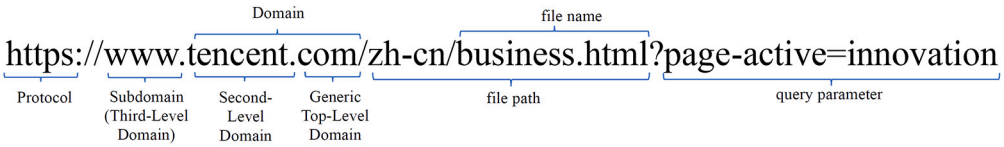

Fig. 7. Structure of a sample URL.

584,909 legitimate samples. The phishing samples were downloaded from two sources: PhishTank open database (OpenDNS, 2022) and Reasonable Antiphishing database (Reasonable, 2021). By sequentially: (1) removing duplicate samples; (2) deleting invalid samples; (3) deleting URLs of defective webpages or "Error 404" pages, 587,668 phishing samples were obtained in DATA2. The legitimate URLs were collected from two sources: the top one million legitimate websites ranked by Alexa (2019) on November 07, 2019, and the Malicious Webpage Analysis Exercise Dataset (China Network Security Technical Challenge, 2017) collected by the 2017 China Network Security Technical Challenge (CNSTC). The following approach was sequentially adopted: (1) retaining the first 500 thousand samples of Alexa; (2) removing duplicate samples; (3) deleting invalid samples; (4) deleting URLs of defective webpages or "Error 404" pages; (5) removing URLs with the file type suffixes (.pdf, .zip, .mp4). Consequently, 584,909 legitimate samples were obtained from DATA2.

In this study, phishing detection was represented as a two-way problem (phishing and legitimate detection). Therefore, phishing websites are considered positive samples, whereas legitimate websites are considered negative samples. The performance of the PDHF was evaluated using a confusion matrix, in which the predicted and actual values were compared. According to the confusion matrix, four metrics, namely *accuracy*, *precision*, *recall*, and F_1-*score*, can be defined to comprehensively evaluate the phishing detection performance.

6.2. Effectiveness of the optimal artificial feature selection

In this section, phishing features that are related to the study at hand are discussed first. Subsequently, the effectiveness of removing the *InVa* irrelevant features is evaluated. Third, the effectiveness of removing redundant SA-based bidirectional features is evaluated. Finally, the performance of the optimal artificial features is assessed.

6.2.1. Phishing features

The features involved in machine learning-based phishing detection systems can be classified into three main categories: URL, HTML, and external features. The public dataset DATA1 provides integral page content, URL and HTML features for researchers to extract and analyze. External features can be obtained by querying third-party services, such as domain registration, search engines, and WHOIS records. As external resources are highly dependent on third parties, and there is a risk of instability in obtaining external resources (for example, the frequently changing results on search engines) (Xiang et al., 2011), external features were not incorporated into the feature set used in this study. Based on existing phishing detection models, 26 URL and 22 HTML features (as listed in Table 3 and Table 4) were collected to represent phishing attacks.

- *URL features*. As shown in Fig. 7, a typical URL is composed of four components: the URL protocol, domain name, file path, and query parameters. Phishing attackers typically disguise illegal URLs as original normal URLs to lure users. The commonly used disguise methods include changing the characters in the original URL to familiar ones, encoding the URLs, and appending other characters after the original characters. For instance, the character 'o' in the original legal URL can be replaced with '0'; characters '.', '@', and '_' can be appended after the original legal characters. Therefore, it is necessary to collect the URL features to recognize phishing websites. As listed in Table 3, 26 URL features were collected for the PDHF model. In this table, each feature is assigned a number. For instance, the feature number of "NumDots" is f_1. It counts the number of "." in the target URL. Among the 26 identified URL features, $f_1, f_2, f_3, f_4, f_5, f_6, f_9, f_{10}, f_{11}, f_{12}, f_{13}, f_{14}, f_{21}, f_{22}, f_{23}$ and f_{25} are integer variables with values greater than or equal to zero; $f_7, f_8, f_{15}, f_{16}, f_{17}, f_{18}, f_{19}, f_{20}, f_{24}$ and f_{26} are Boolean variables with values of zero or one.
- *HTML features*. Phishing websites are often designed to resemble legitimate websites to lure users. On websites, network resources such as pictures, videos, and formatted texts, can be displayed by HTML labels. Many phishing attacks can be detected by analyzing the HTML structural features. As listed in Table 4, by referring to the existing research on phishing website detection based on machine learning, such as studies (Silva et al., 2020), (Chiew et al., 2019) and (Zhu et al., 2020), 22 HTML features were collected for the PDHF model. For instance, the feature number of "PctExtHyperlinks" is f_{27}. In the HTML source code of a webpage, f_{27} is the percentage of hyperlinks pointing to outside of this webpage in all links of this webpage. Among the 22 identified HTML features, f_{27}, f_{28} and f_{34} are float variables with values greater than or equal to zero; $f_{29}, f_{30}, f_{31}, f_{32}, f_{33}, f_{35}, f_{36}, f_{37}, f_{38}, f_{39}, f_{40}, f_{41}$ and f_{42} are Boolean variables with values of zero or one; $f_{43}, f_{44}, f_{45}, f_{46}, f_{47}$ and f_{48} are categorial variables which apply rules and thresholds to generate values containing zero and one (Tan, 2018).

6.2.2. Effectiveness of the removal of InVa irrelevant features

In this section, DATA1 is used to evaluate the effectiveness of the methods for removing irrelevant features. After *WoE* encoding, the *InVa* values of the 48 features (listed in Tables 3 and 4) were calculated. As shown in Fig. 8, the 48 combined URL and HTML features are in descending order according to *InVa* values. In this figure, only feature numbers (first column of Tables 3 and 4) are listed to save space.

Fig. 9 shows the detection accuracy of different classifiers by continuously appending important (high *InVa* value) features. The order in which the features are added is based on the descending order of sorting of the *InVa* values shown in Fig. 8. In this experiment, to evaluate the proposed method more accurately, we averaged 10 experimental data

Table 3
Identified URL features.

No.	Feature name	Description	Value type
f_1:	NumDots	Number of dots in the URL	int
f_2:	SubdomainLevel	Level of subdomain in the URL	int
f_3:	PathLevel	Depth of the path in the URL	int
f_4:	UrlLength	Total number of characters in the URL	int
f_5:	NumDash	Number of '-' in the URL	int
f_6:	NumDashInHostname	Number of '-' in hostname part of the URL	int
f_7:	AtSymbol	If '@' exists in the URL	bool
f_8:	Tidesymbol	If '~' exists in the URL	bool
f_9:	NumUnderscore	Number of '_' in the URL	int
f_{10}:	NumPercent	Number of '%' in the URL	int
f_{11}:	NumQueryComponents	Number of query parts in the URL	int
f_{12}:	NumAmpersand	Number of '&' in the URL	int
f_{13}:	NumHash	Number of '#' in the URL	int
f_{14}:	NumNumericChars	Number of numeric characters in the URL	int
f_{15}:	NoHttps	If HTTPS exist in the URL	bool
f_{16}:	RandomString	If random strings exist in the URL	bool
f_{17}:	IpAddress	If IP address is used in the hostname part of the URL	bool
f_{18}:	DomainInSubdomains	If TLD or ccTLD is used as part of subdomain in the URL	bool
f_{19}:	DomainInPaths	If TLD or ccTLD is used in the path of in the URL	bool
f_{20}:	HttpsInHostname	If HTTPS in obfuscated in the hostname part of the URL	bool
f_{21}:	HostnameLength	Total characters in the hostname part of the URL	int
f_{22}:	PathLength	Total characters in the path of the URL	int
f_{23}:	QueryLength	Total characters in the query part of the URL	int
f_{24}:	DoubleSlashInPath	If '//' exists in the path of the URL	bool
f_{25}:	NumSensitiveWords	Number of sensitive words (i.e., "secure", "account", ...) in the URL	int
f_{26}:	EmbeddedBrandName	If brand name appears in subdomains and path of the URL	bool

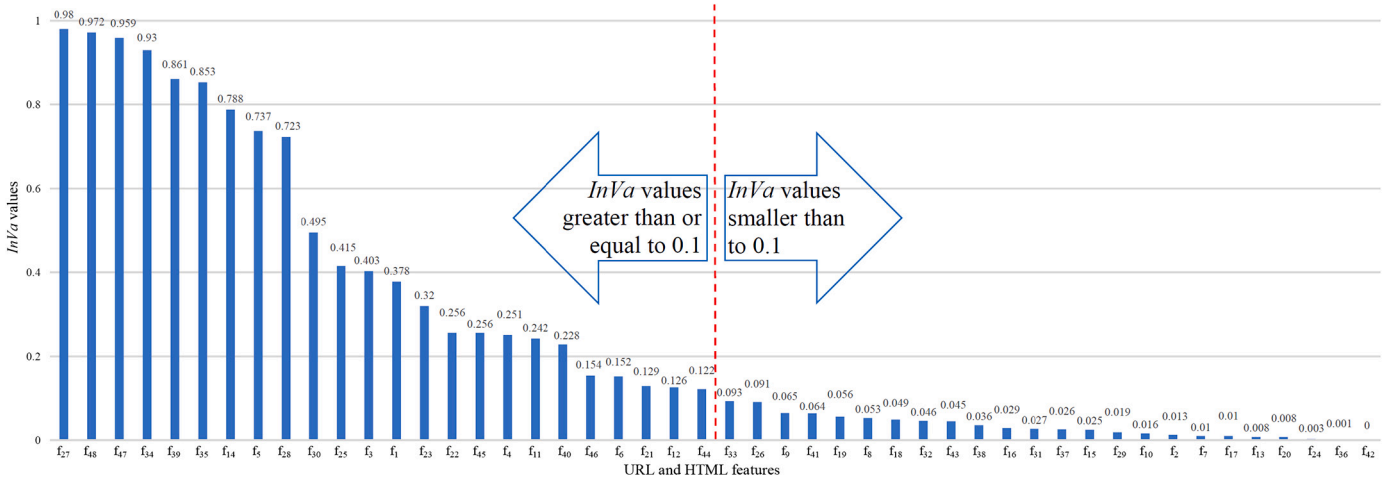

Fig. 8. *InVa* values of the 48 identified HTML and URL features for DATA1 (in descending order based on *InVa* values).

points as the result. Five commonly used classifiers were selected: Logistic Regression, Support Vector Machine, Naive Bayes, Decision Tree, and Random Forest.

As shown in Fig. 9, in the initial stage, the feature set only contains f_{27}. The accuracies of the five classifiers are 0.6115, 0.8489, 0.6453, 0.9182, and 0.9188 in sequence, in the (2^{nd} column of Table 5). In this figure, "*Accuracy*" is a cumulative value as features are added. For example, as shown in this figure, the "f_{35}" on the horizontal axis represents the feature subset of $\{f_{27}, f_{48}, f_{47}, f_{34}, f_{39}, f_{35}\}$. The accuracy of this subset on Logistic Regression is 78.1%. For all the classifiers, accuracy increases with an increase in the number of important *InVa* features. However, the improvement in accuracy of each classifier tends to stabilize when the number of added features is half of the original features.

Based on the experimental results, we set a cut-off value of 0.1 and removed features below this value as irrelevant features. Furthermore, we selected features with *InVa* values higher than 0.1 for testing, and the experimental results in Table 5 verify the rationality of selecting 0.1 as the cut-off value. The experimental results also indicate that features with values below 0.1 have almost no impact on improving accuracy.

In this study, 24 features with *InVa* values smaller than 0.1 (the rightmost 24 features in Figs. 8 and 9) were considered irrelevant. Consequently, the 24 high *InVa* features were selected (the leftmost 24 features in Figs. 8 and 9) f_{27}(0.980), f_{48}(0.972), f_{47}(0.959), f_{34}(0.930), f_{39}(0.861), f_{35}(0.853), f_{14}(0.788), f_5(0.737), f_{28}(0.723), f_{30}(0.495), f_{25}(0.415), f_3(0.403), f_1(0.378), f_{23}(0.320), f_{22}(0.256), f_{45}(0.256), f_4(0.251), f_{11}(0.242), f_{40} (0.228), f_{46}(0.154), f_6(0.152), f_{21}(0.129), f_{12}(0.126), and f_{44}(0.122).

Table 5 lists the detection accuracy comparisons of the different classifiers among the 24 high *InVa* and 48 features. In this table, the 3^{rd} and 4^{th} columns list the cumulative accuracy for all 24 and 48 features combined, respectively. All classifiers reduce the original feature size by 50% without sacrificing phishing detection accuracy. Moreover, in comparing stability (Fig. 9) and accuracy (Table 5), the Random Forest classifier is superior to the other classifiers. This finding is consistent with those of previous studies (Chiew et al., 2019)(Sahingoz et al.,

Table 4
Identified HTML features.

No.	Feature name	Description	Value type
f_{27}:	PctExtHyperlinks	Percentage of external hyperlinks in HTML source code	float
f_{28}:	PctExtResourceUrls	Percentage of external resource URLs in HTML source code	float
f_{29}:	ExtFavicon	"Favicon" loaded from a domain name different from the URL domain name	bool
f_{30}:	InsecureForms	If the form action attribute contains a URL without HTTPS protocol	bool
f_{31}:	RelativeFormAction	If the form action attribute contains a relative URL	bool
f_{32}:	ExtFormAction	If the form action attribute contains a URL from an external domain	bool
f_{33}:	AbnormalFormAction	If the form action attribute contains '#'/"about:blank"/empty string/"javascript:true"	bool
f_{34}:	PctNullSelfRedirect-Hyperlinks	Percentage of hyperlinks containing empty value, self-redirect value, the URL of current page, or abnormal value (such as "file://E:/")	float
f_{35}:	FrequentDomainName-Mismatch	If the most frequent domain name in HTML source code does not match the webpage URL domain name	bool
f_{36}:	FakeLinkInStatusBar	If HTML code uses command "onMouseOver" to display fake URL in the status bar	bool
f_{37}:	RightClickDisabled	If HTML code uses JavaScript command to disable right click operation	bool
f_{38}:	PopUpWindow	If HTML code uses JavaScript command to launch "pop-ups"	bool
f_{39}:	SubmitInfoToEmail	If HTML source code contains the HTML "mailto" function	bool
f_{40}:	IframeOrFrame	If iframe or frame is used in HTML source code	bool
f_{41}:	MissingTitle	If the title tag is empty in HTML source code	bool
f_{42}:	ImagesOnlyInForm	If the form scope in HTML code contains no text at all but only images	bool
f_{43}:	SubdomainLevelRT	Number of dots in hostname part of URL.	Categorial[1]
f_{44}:	UrlLengthRT	Number of total characters in the webpage URL	Categorial
f_{45}:	PctExtResourceUrlsRT	Percentage of external resource URLs in HTML source code	Categorial
f_{46}:	AbnorExtFormActionR	If the form action attribute contains a foreign domain/"about:blank"/an empty string	Categorial
f_{47}:	ExtMetaScriptLinkRT	Percentage of meta, script, and link tags containing external URL in the attributes	Categorial
f_{48}:	PctExtNullSelfRedirect-HyperlinksRT	Percentage of hyperlinks in HTML source code that use different domain names, starts with '#', or using "JavaScript::void(0)"	Categorial

[1] Categorial: Apply rules and thresholds to generate value (Tan, 2018).

Table 5
Accuracy comparisons of five classifiers on different feature sets.

Models	Initial feature (f_{27})	24 high *InVa* features	48 collected artificial features
Logistic Regression	0.6115	0.9220	0.9324
Support Vector Machine	0.8489	0.8805	0.8681
Naive Bayes	0.6453	0.8550	0.8555
Decision Tree	0.9182	0.9650	0.9660
Random Forest	**0.9188**	**0.9745**	**0.9774**

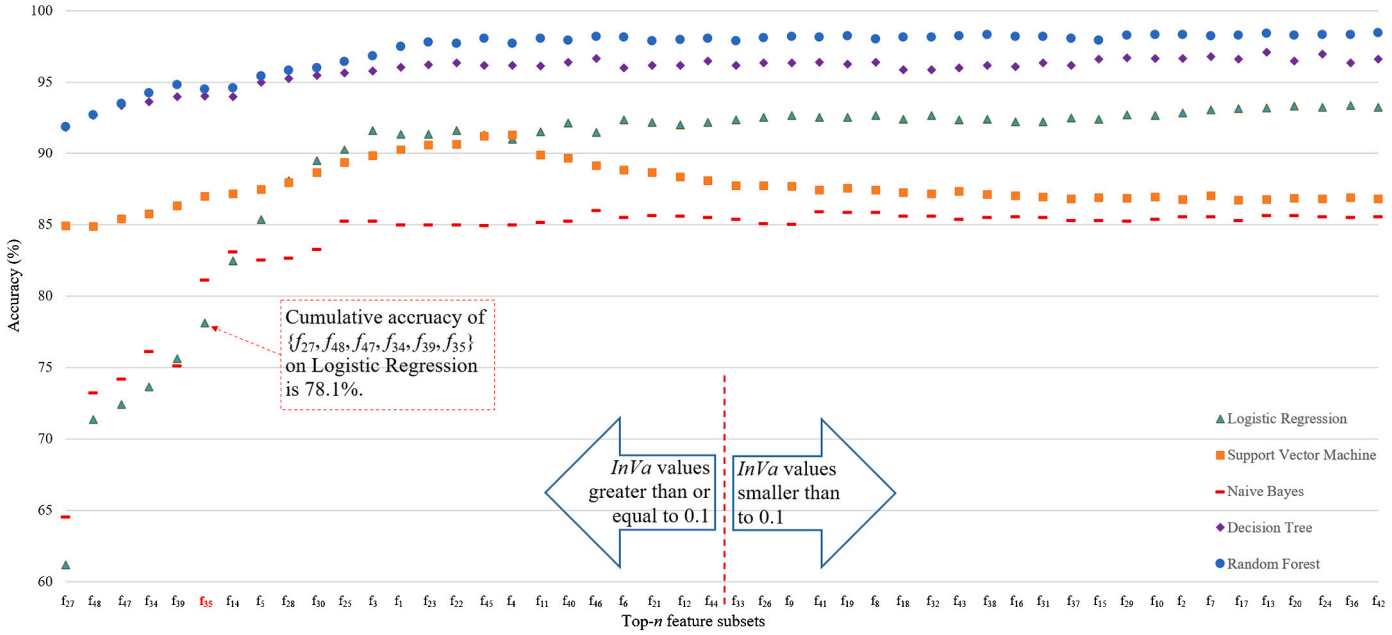

Fig. 9. Detection accuracy of five classifiers after sequentially appending important features.

Table 6
Number of optimal features and accuracy of different search approaches.

Approaches	Number of features	Selected features	Accuracy
Forward search	21	f_1, f_3, f_5, f_{11}, f_{12}, f_{14}, f_{22}, f_{23}, f_{25}, f_{27}, f_{28}, f_{30}, f_{34}, f_{35}, f_{39}, f_{40}, f_{44}, f_{45}, f_{46}, f_{47}, f_{48}	0.9747
Backward search	23	f_1, f_3, f_4, f_5, f_6, f_{11}, f_{12}, f_{14}, f_{21}, f_{22}, f_{23}, f_{25}, f_{27}, f_{28}, f_{30}, f_{34}, f_{35}, f_{39}, f_{40}, f_{45}, f_{46}, f_{47}, f_{48}	0.9749
Bidirectional Search	20	f_1, f_3, f_4, f_5, f_{12}, f_{14}, f_{22}, f_{23}, f_{25}, f_{27}, f_{28}, f_{30}, f_{34}, f_{35}, f_{39}, f_{40}, f_{45}, f_{46}, f_{47}, f_{48}	0.9753
SA-based bidirectional search	18	f_1, f_3, f_5, f_{12}, f_{14}, f_{23}, f_{25}, f_{27}, f_{28}, f_{30}, f_{34}, f_{35}, f_{39}, f_{40}, f_{45}, f_{46}, f_{47}, f_{48}	**0.9769**

2019). Therefore, Random Forest was selected as the underlying classifier for our phishing detection model.

6.2.3. *Effectiveness of the removal of SA-based bidirectional redundant features*

Experiments were conducted to evaluate the effectiveness of the SA-based bidirectional search algorithm, thereby removing redundant interrelated features. In this experiment, the 24 features selected in the previous section were used as initial feature sets. As Random Forest exhibits excellent classification performance, it was used as the underlying classifier. The parameters T_0, T_F, α, and N used in the SA-based bidirectional search method were set to 100, 0.1, 0.95, and 1000, respectively.

Using the proposed SA-based bidirectional search method, 18 optimal artificial features were selected (as listed in the 5^{th} row and 3^{rd} column of Table 6). To verify the performance of the proposed SA-based bidirectional search method, three other methods, namely forward search, backward search, and bidirectional search, were also evaluated.

Table 6 lists the selected features and accuracies obtained using the different search approaches. In this table, the 4^{th} column lists the cumulative accuracy when the corresponding features listed in the 3^{rd} column are combined. As indicated in the table, the proposed SA-based bidirectional search method achieves the highest detection accuracy with the smallest number of features. This indicates that the randomized strategy of SA can help the feature search procedure jump out of the local optima, and a set of optimal features can be obtained by combining this strategy with a bidirectional search.

6.2.4. *Performance of the final optimal artificial features*

Table 7 lists performance comparisons of the initial features (the total 48 features listed in Tables 3 and 4), 24 high *InVa* features (obtained by removing the *InVa* irrelevant features from the initial features), and 18 optimal features (obtained by the SA-based bidirectional search from the 24 high *InVa* features) in terms of accuracy, time cost, and normalized performance score (*Accuracy/Time cost*).

Because the public dataset DATA1 provides integral page content, URL, and HTML features for researchers to extract and analyze, this experiment was conducted on DATA1. In this experiment, the K-fold cross-validation method was used to evaluate classification accuracy, time cost, and normalized performance score, which were collected and calculated sequentially. In the K-fold cross-validation method, the value of K was set to 10; that is, DATA1 was divided into 10 subsets, of which nine subsets constituted the training sets and one subset was set aside for the testing set. The 500 phishing and 500 legitimate samples in the testing set were randomly selected from DATA1. The time costs listed in the fourth column of this table are the average values on 1000 testing samples obtained at the testing stage of RF.

According to the results summarized in Table 7, compared with the *initial features set*, the *optimal features set* reduces the feature size by 62.5% at the expense of reducing accuracy by only 0.05%. Because of the small number of calculations, the *optimal feature set* consumes less time than the other two feature sets. The low time overhead allows the proposed method to detect phishing in real-time. Moreover, the *optimal features set* obtains the highest normalized performance score (the higher the value, the better the performance). The high detection accuracy, high normalized performance score, and low time cost

Table 7
Performance comparisons among different feature sets.

Feature set	Feature number	Accuracy	Time cost (ms)	Normalized performance score
Initial features	48	**0.9774**	0.0239	40.8954
High *InVa* features	24	0.9745	0.0172	56.6570
Optimal features	18	0.9769	**0.0160**	**61.0563**

Table 8
Phishing detection performance of traditional models on the two datasets.

Model	Accuracy		Precision		Recall		F_1-score	
	DATA1	DATA2	DATA1	DATA2	DATA1	DATA2	DATA1	DATA2
Logistic Regression	0.9230	0.8673	0.9175	0.8526	0.9232	0.8360	0.9204	0.8350
Support Vector Machine	0.9185	0.8596	0.9074	0.8893	0.9357	0.8914	0.9214	0.8903
Naive Bayes	0.8540	0.8208	0.9335	0.8534	0.7282	0.8184	0.8182	0.8153
Decision Tree	0.9675	0.9521	0.9553	0.9577	0.9796	0.9562	0.9673	0.9564
Random Forest	**0.9769**	**0.9612**	**0.9766**	**0.9623**	**0.9823**	**0.9607**	**0.9794**	**0.9615**

demonstrate that the combination of the *InVa* index and SA-based bidirectional search is effective in generating optimal features.

Using the 18 optimal features, the performances of different classifiers on DATA1 and DATA2 were tested. Considering the large size of DATA2, using K-fold cross-validation similar to DATA1 would consume significant time and computational resources. Thus, 80%, 10%, and 10% of DATA2 samples were used for training, verification, and testing, respectively.

Table 8 summarizes the experimental results. Through comparisons, it was established that the detection accuracies of the different classifiers for DATA2 are generally lower than those for DATA1. This difference indicates that a machine learning detection model based on artificial features has an overfitting problem when processing large-scale datasets. This may also be caused by avoidance measures taken after artificially extracted features have been adapted by phishing attackers. To remedy this defect, automatic phishing feature learning approaches are urgently required to compensate for manually extracted features.

6.3. Performance of the CNN-DQA

The hyperparameters of the CNN used in the CNN-DQA model were trained and experimentally determined. Subsequently, the performance of the constructed CNN combined with a disorderly quantized attention mechanism was evaluated. Deep-learning methods can automatically extract useful features by training on massive amounts of data. To obtain high-quality deep features, the large-scale DATA2 was used to train the CNN-DQA model. For this dataset, 80%, 10%, and 10% of the samples were used for training, verification, and testing, respectively. The training set was used to train the weight, deviation, and other parameters of the CNN; the verification set was used to determine the hyperparameters of the neural network; and the testing set was used to evaluate the overall performance of CNN-DQA.

6.3.1. Hyperparameters of the CNN

To obtain the hyperparameters of the CNN, the dimension of the embedding layers was set to 128, and the sizes of the convolution kernels q_1 and q_2 were set to three and five, respectively. The batch size, i.e., the number of URLs that the model can handle at a time, was set to 256. To prevent overfitting, the early stop method was used to train the model.

- *Length of URL*. Because the CNN in the CNN-DQA model can only deal with fixed length vectors, the length of the URLs was fixed to L with all excess characters deleted and vacant spots filled with the "pad" characters. To extract high-quality phishing features, it is necessary to determine the value of L.
  The first two columns, *URL length* and *Count*, of Table 9 specify the URL distributions for different lengths in DATA2. In the URL distribution in the table, the lengths of the URLs vary significantly; however, most of them are not longer than 200. Therefore, the performances of the CNN with L ranging from 20 to 200 were tested. In this experiment, the number of feature maps corresponding to each convolution kernel was set to 64.
  The performance (*Accuracy*, *Precision*, *Recall*, and F_1-*score*) of the CNN for different lengths of URLs is summarized in Table 9. As presented in the table, the CNN has the worst performance (with only 91.08% *Accuracy*) when L is set to 20. Generally, shorter URLs lack sufficient information; thus, the CNN cannot sufficiently learn effective features. As the lengths of the URLs increase, the performance of CNN improves. However, as the value of L continues to increase, the performance of CNN shows a downward trend. To confuse users, phishing URLs are extremely long and contain many useless and redundant information. If CNN learns too many useless or redundant features, its accuracy in detecting phishing URLs declines. Therefore, the L of the URLs cannot be too short or too long. It is observed in the table that, when L ranges from 80 to 140, the *Accuracy*, *Precision*, *Recall*, and F_1-*score* of CNN exceed 98%. The performance of the CNN is the best when L is set to 100. Therefore, L in the CNN of the CNN-DQA was set to 100.
- *Number of feature maps per convolution kernel*. Different numbers of feature maps (logarithmic scale) in the convolution kernel may affect the performance of the underlying CNN differently. Here, the appropriate number of feature maps per convolution kernel was tuned for the length of the URL set to 100. Table 10 summarizes the performance of the underlying CNN with different numbers of feature maps for each convolution kernel. The data listed in the table indicate that the greater the number of feature maps, the better is the performance of the underlying CNN model. However, the performance improvement decreases as the number of feature maps reach a specific value. Moreover, with more feature maps, a large amount of time is required to train the underlying model.
  Fig. 10 shows the relationship between accuracy and the number of feature maps. As shown in this figure, the accuracy increases with a continuous increase in the number of feature maps. When the number of feature maps is greater than 64, the improvement trend of the accuracy tends to be flat. That is, more resources cannot achieve better returns. Therefore, the number of feature maps for each convolution kernel was set to 64 to balance the performance and computational resource overhead.

6.3.2. Effectiveness of the disorderly quantized attention mechanism

To reduce complexity and accelerate convergence speed, a disorderly quantized attention mechanism was integrated into the phishing website detection model. In CNN-DQA, important features are selected

Table 9
Length distribution of URLs in DATA2 and the impact of different L values on CNN performance.

URL length	Count		L	Accuracy	Precision	Recall	F_1-score
	Phishing	Legal					
[0,20)	13555	45395	20	0.9108	0.9247	0.9055	0.9150
[20,40)	129034	515251	40	0.9282	0.9435	0.9287	0.9360
[40,60)	155257	17533	60	0.9678	0.9664	0.9621	0.9642
[60,80)	107130	5033	80	0.9802	0.9795	0.9777	0.9786
[80,100)	56497	730	100	0.9828	0.9847	0.9813	0.9825
[100,120)	30396	324	120	0.9822	0.9856	0.9817	0.9836
[120,140)	18740	201	140	0.9817	0.9843	0.9806	0.9824
[140,160)	14907	165	160	0.9793	0.9809	0.9807	0.9808
[160,180)	9647	73	180	0.9786	0.9797	0.9742	0.9769
[180,200)	7005	58	200	0.9755	0.9826	0.9733	0.9779

Table 10
CNN performance for different number of feature maps per convolution kernel.

Feature Maps	Accuracy	Precision	Recall	F_1-score
4	0.9569	0.9582	0.9498	0.9540
8	0.9658	0.9664	0.9616	0.9640
16	0.9752	0.9744	0.9731	0.9738
32	0.9799	0.9785	0.9762	0.9773
64	0.9828	0.9847	0.9803	0.9825
128	0.9831	0.9833	0.9825	0.9829
256	0.9832	0.9826	0.9817	0.9822

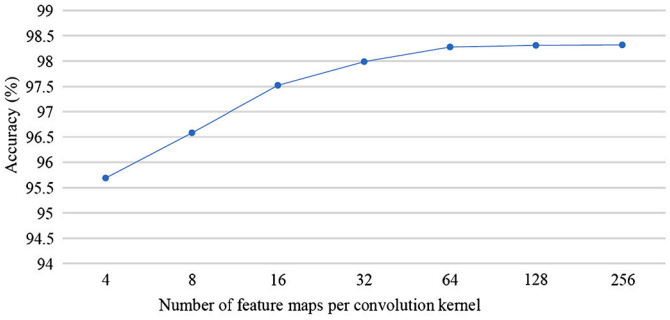

Fig. 10. Relationship between accuracy and the number of feature maps.

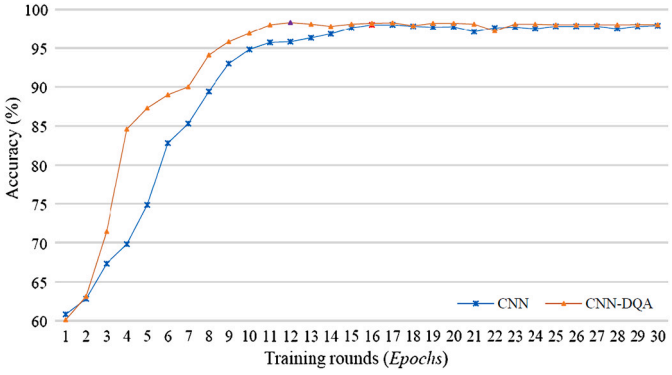

Fig. 11. Training processes of CNN and CNN-DQA.

from the original features generated by the CNN, and the focus is only on the features that contribute more to the performance of phishing detection. By assigning high weights to important features and low weights to useless features, CNN-DQA reduces the complexity of the underlying CNN model without sacrificing performance.

In deep learning, the *epoch* index represents the number of rounds required for the model to converge during training in one epoch, that is, all samples in the training set are trained once. Therefore, the convergence speed of the proposed model is effectively described.

Fig. 11 shows the training process of CNN and CNN-DQA on the training set of DATA2. Here, CNN represents a one-dimensional CNN with all automatically generated features, and CNN-DQA is a one-dimensional CNN that integrates the disorderly quantized attention mechanism with refining phishing features. In the figure, CNN-DQA reaches the highest detection accuracy in 12 training rounds, whereas CNN requires 16 training rounds. The CNN-DQA converges faster than CNN.

Table 11 compares the performance of CNN and CNN-DQA on the DATA2 testing set. As shown in the table, refining the features automatically extracted by the CNN, does not degrade the performance of CNN-DQA, and accuracy is improved. Using deep learning technology, CNN-DQA does not require artificially extracted features and does not rely on HTML or third-party features. Therefore, CNN-DQA is highly efficient and accurate, detecting URLs in real time.

6.4. Performance of PDHF

The PDHF phishing detection model is constructed using hybrid features composed of optimal artificial and automatic deep learning features. Based on these hybrid features, the underlying RF classifier was trained and used to detect phishing URLs. First, the fitness of selecting RF as the underlying classifier was evaluated. Thereafter, the performances of $\boldsymbol{F}_{optimal}$, $\boldsymbol{F}_{deep}$, and $\boldsymbol{F}_{hybrid}$ were compared. Finally, the performance of the PDHF and those of the nine existing phishing detection systems were compared. Here, experiments conducted on DATA2 (80%, 10%, and 10% of the samples used for training, verification, and testing, respectively) are described.

6.4.1. Selection of the underlying classifier

Based on the SA-based bidirectional feature search algorithm and CNN-DQA deep model, $\boldsymbol{F}_{optimal}$ and $\boldsymbol{F}_{deep}$ were generated; $\boldsymbol{F}_{hybrid}$ was generated by combining $\boldsymbol{F}_{optimal}$ and $\boldsymbol{F}_{deep}$. Using $\boldsymbol{F}_{hybrid}$, the performance of the five different classifiers was tested on DATA2. The experimental results are listed in Table 12. The superior performance ($Accuracy > 98\%$) of the different classifiers demonstrates the effectiveness of $\boldsymbol{F}_{hybrid}$. Random Forest is better than the other classifiers on all test indices. This finding is consistent with the experimental results dis-

Table 11
Effectiveness of the disorderly quantized attention mechanism.

Models	Accuracy	Precision	Recall	F_1-score
CNN	0.9828	0.9804	0.9792	0.9798
CNN-DAQ	**0.9833**	**0.9830**	**0.9823**	**0.9827**

Table 12
Performances of different classifiers on DATA2.

Classifier	Accuracy	Precision	Recall	F_1-score
Logistic Regression	0.9829	0.9842	0.9833	0.9838
Naive Bayes	0.9837	0.9864	0.9829	0.9846
Support Vector Machine	0.9862	0.9865	0.9855	0.9860
Decision Tree	0.9946	0.9930	0.9918	0.9924
Random Forest	**0.9965**	**0.9942**	**0.9940**	**0.9941**

Table 13
Performance comparisons among different phishing feature sets on DATA2.

Models	Accuracy	Precision	Recall	F_1-score
$\boldsymbol{F}_{optimal} + RF$	0.9612	0.9623	0.9607	0.9615
$\boldsymbol{F}_{deep} + RF$	0.9838	0.9831	0.9822	0.9826
CNN-DQA	0.9833	0.9830	0.9823	0.9827
$\boldsymbol{F}_{hybrid} + RF$(PDHF)	**0.9965**	**0.9942**	**0.9940**	**0.9941**

cussed in Section 6.2. Therefore, Random Forest was selected as the underlying classifier in our phishing detection model.

6.4.2. Performance comparisons among different phishing feature sets

The performances of $\boldsymbol{F}_{optimal}$, $\boldsymbol{F}_{deep}$, and $\boldsymbol{F}_{hybrid}$ are compared using Random Forest as the underlying classifier. In Table 13, $\boldsymbol{F}_{optimal}$+RF, $\boldsymbol{F}_{deep}$+RF, and $\boldsymbol{F}_{hybrid}$+RF (PDHF) represent the performances of Random Forest combined with corresponding feature vectors ($\boldsymbol{F}_{optimal}$, $\boldsymbol{F}_{deep}$, and $\boldsymbol{F}_{hybrid}$), respectively. The performance of the CNN-DQA is also listed in this table.

As indicated by the 96.12% phishing detection accuracy listed in this table, relying solely on artificial phishing features is insufficient. Automatic deep features, such as the data of $\boldsymbol{F}_{deep}$ and CNN-DQA, can achieve good phishing detection performance. This can be used to supplement artificial features as with $\boldsymbol{F}_{hybrid}$+RF, that is, the final form of the PDHF model, which exhibits the best performance. Meanwhile, the relatively small size of the final trained PDHF model (94 MB) and low time consumption on each URL (35.6 ms on average) make it suitable as a security plug-in for clients, browsers, and various instant messaging tools that run on different platforms.

6.4.3. Overall performance comparisons among different phishing detection systems

In this section, the overall performance of the PDHF is compared with those of the 12 existing phishing detection models. Table 14 lists the phishing detection performance of these models. The last column lists the average time cost of phishing detection for these models.

In this table, the data pertaining to the first six models and PDHF (as indicated in the final row) represent the results derived from our experimental investigations conducted on DATA2. Owing to the dynamic nature of various websites, including the possibility of some websites no longer being accessible, and the variability in network accessibility across different regions, the content and code associated with the target websites during the testing phase remain inherently unpredictable. To ensure fairness in testing, from the DATA2, we selected 1000 URLs and stored their relevant code on the same machine where test models were installed. Meanwhile, due to different classification methods, the training time of different models ranges from tens of minutes to over ten hours. In this experiment, the time costs listed in Table 14 are the average values of 1000 samples of DATA2 at the testing stage of different models.

The 30 initial artificial features of HS (Babagoli et al., 2019) and DTOF-ANN (Zhu et al., 2020) are collected from the address bar, abnormal behavior of the source codes, HTML and JavaScript, and domain names. To enhance the efficiency of the phishing detection model, HS employs a combination of decision tree and wrapper approaches, ultimately selecting 20 optimal features. It is important to note that, due to the application of nonlinear regression techniques and Support Vector Machines (SVM), HS incurs a significantly higher computational time cost compared to the other five models under consideration. Conversely, DTOF-ANN employs a feature selection process that involves the use of the Gini coefficient and a feature evaluation index based on decision trees, culminating in the selection of 14 optimal features from the initial feature set. Notably, owing to the relatively small number of selected optimal features and the underlying Artificial Neural Network (ANN) architecture, DTOF-ANN achieves a considerably reduced computational time cost in the context of phishing detection. As discussed in this article, the extracted features may be adapted by phishing attackers for bypassing detection. Their phishing detection performance (accuracy, precision, recall, and F_1-score) is worse than the other models.

LightGBM (Oram et al., 2021) is a light gradient boosting machine-based phishing email detection model. The dataset used for building LightGBM is DATA1 which originally contains 48 features. By merging mutually exclusive features and balancing dataset based on random naive over-sampling, LightGBM exhibits high detection performance and low time cost. CyberLen (Liang et al., 2022) is a deep learning-based system for the robust and effective detection of phishing URLs. In CyberLen, lexical features and deep structural features are learned at first. Then, a self-paced wide & deep learning strategy is proposed to fuse heterogeneous features. CyberLen reduces the computational complexity by exploring the deeper network architecture and the parallel computation of the temporal convolution network (TCN).

MOE/RF (Zhu et al., 2022) is a feature-based phishing detection model that balances accuracy and recall. Notably, MOE/RF adopts a feature selection approach that does not prescribe a particular category of initial features. In this experiment, the 48 features listed in Tables 3 and 4 are set as the initial feature set for MOE/RF. Due to the multi-objective evolution algorithm for finding optimal solutions

Table 14
Performances among different phishing detection models.

Models	Datasets		Accuracy	Precision	Recall	$\mathbf{F}_1$-score	Time cost (*ms*)
	Legal	Phishing					
HS (2015) (Babagoli et al., 2019)	584909	587668	0.9667	0.9705	0.9741	0.9695	85.3
DTOF-ANN (2020) (Zhu et al., 2020)	584909	587668	0.9778	0.9740	0.9739	0.9752	**16.7**
LightGBM (2021) (Oram et al., 2021)	584909	587668	0.9833	0.9822	0.9872	0.9815	18.9
MOE/RF (2022) (Zhu et al., 2022)	584909	587668	0.9851	0.9860	0.9842	0.9854	41.2
CyberLen (2022) (Liang et al., 2022)	584909	587668	0.9869	0.9752	0.9462	0.9561	22.6
WebPhish (2024) (Opara et al., 2024)	584909	587668	0.9824	0.9827	0.9818	0.9820	24.1
CANTINA+ (2011) (Xiang et al., 2011)	1868	940	0.9550	0.9640	0.9640	0.9640	–
PhishStorm (2014) (Marchal et al., 2014)	48009	48009	0.9491	0.9844	0.9127	0.9472	–
Off-the-hook (2017) (Marchal et al., 2017)	20000	2000	**0.9990**	0.9750	0.9510	0.9630	–
Phishing-Aware (2018) (Pham et al., 2018)	10000	11600	0.9836	0.9852	0.9910	0.9881	–
CNN-MHSA (2020) (Xiao et al., 2020)	45000	43984	0.9862	0.9816	0.9844	0.9830	–
CrawlPhish (2022) (Zhang et al., 2022)	2000	2000	0.9840	0.9825	0.9855	0.9840	–
PDHF	584909	587668	0.9965	**0.9942**	**0.9940**	**0.9941**	35.6

and the underlying Random Forest classifier, MOE/RF incurs a greater computational time cost compared to the other models.

WebPhish (Opara et al., 2024) represents a sophisticated deep neural network model designed to identify phishing attacks by analyzing raw URLs and HTML content. Due to adopting a similar method, as indicated in the last line of Table 11, the performance of WebPhish and CNN-DQA exhibits a closely comparable level of effectiveness. The optimal artificial feature selection along with CNN combined with the attention mechanism, has enhanced the performance metrics, including accuracy, precision, recall, and F_1-score, of PDHF when compared to those achieved by single-machine and single deep learning techniques. However, because of the final feature vector and the utilization of a Random Forest classifier, PDHF does not attain the highest level of computational efficiency among the seven models assessed on the DATA2 dataset.

Different technologies, the short lifecycle of phishing websites, and third-party-supported feature selection, make it difficult to re-implement the algorithms of all the existing models. As there were significant differences between the experimental results (obtained by our experiments) and the original data in the literature first published, the data of the remaining 6 models were directly obtained from the corresponding references. Consequently, the time cost of these models is not listed. Among the 6 remaining models, CANTINA+ (Xiang et al., 2011) is a classical machine-learning phishing detection framework based on eight self-defined features borrowed from the HTML document object model, search engines, and third-party services; PhishStorm (Marchal et al., 2014) is an automated phishing detection system that analyzes URLs in real time to identify potential phishing sites; Off-the-hook (Marchal et al., 2017) is a machine learning phishing detection framework that extracts more than 200 features; the Phishing-Aware (Pham et al., 2018) model can transparently monitor and protect fog users from phishing attacks; CNN-MHSA (Xiao et al., 2020) is a combined CNN and multi-head self-attention (MHSA) approach for detecting phishing websites; and CrawlPhish (Zhang et al., 2022) is a framework for automatically detecting phishing websites based on visual and code similarities. As shown in Table 14, PDHF exhibits higher performance than the other models, except for off-the-hook. With 210 features from URLs and relevant websites, Off-the-hook has the highest phishing detection accuracy. However, it exhibits relatively lower performance on the other three indices.

7. Conclusion and future work

This study proposes PDHF, which is a novel phishing detection model based on a combination of optimal artificial and automatic deep features. To obtain optimal artificial features, initially, the *InVa* feature importance evaluation index, which is based on *WoE* feature encoding, was designed to eliminate irrelevant features. Thereafter, a bidirectional search algorithm, which was revised using a simulated annealing randomization strategy, was designed to delete the redundant features. To extend the timeliness and further improve the performance of phishing detection, the CNN-DQA model, which is based on a one-dimensional character CNN and a disorderly quantized attention mechanism, was designed to automatically learn features from the target URLs or to detect phishing URLs separately. Finally, the PDHF phishing detection model was built based on the RF classifier by combining optimal artificial and deep URL features. The experimental results from testing on two phishing datasets with different scales demonstrated that PDHF performs better than several existing methods. Currently, the CNN-DQA proposed in this study can only automatically learn deep features from URLs of phishing websites. To further improve phishing detection performance, in future work, effective methods to automatically learn deep features from structurally complex page content and HTML source code will be developed. Furthermore, to ensure user data security, the proposed phishing detection method will be used as a security plug-in for clients, browsers, and various instant messaging tools that run on network edges, personal computers, mobile phones, and other personal terminals.

CRediT authorship contribution statement

Erzhou Zhu: Conceptualization, Funding acquisition, Methodology, Project administration, Resources, Supervision, Writing – review & editing. **Kang Cheng:** Software, Writing – original draft. **Zhizheng Zhang:** Investigation, Software, Writing – original draft. **Huabin Wang:** Data curation, Formal analysis, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This study was supported by the Natural Science Foundation of Anhui Province (China) (No. 2008085MF188, 1908085MF209) and the

University Natural Science Research Project of Anhui Province (China) (No. KJ2021A0041).

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cose.2023.103561.

References

Abdelnabi, S., Krombholz, K., Fritz, M., 2020. VisualPhishNet: zero-day phishing website detection by visual similarity. In: Proc. of the CCS 2020, pp. 1681–1698.

Akiyama, M., Yagi, T., Itoh, M., 2011. Searching structural neighborhood of malicious URLs to improve blacklisting. In: Proc. of the SAINT 2011, pp. 1–10.

Alexa, 2019. Top one million legitimate websites [Online]. http://www.seobook.com/download-alexa-top-1-000-000-websites-free.

Almomani, A., Alauthman, M., Shatnawi, M.T., Alweshah, M., Alrosan, A., Alomoush, W., Gupta, B.B., 2022. Phishing website detection with semantic features based on machine learning classifiers: a comparative study. Int. J. Semantic Web Inf. Syst. 18 (1), 1–24. https://doi.org/10.4018/IJSWIS.297032.

APWG, 2022. Phishing activity trends report: 4th quarter 2022 [Online]. https://docs.apwg.org/reports/apwg_trends_report_q4_2022.pdf.

Azeez, N.A., Misra, S., Margaret, I.A., Sanz, L.F., Abdulhamid, S.M., 2021. Adopting automated whitelist approach for detecting phishing attacks. Comput. Secur. 108, 102328. https://doi.org/10.1016/j.cose.2021.102328.

Babagoli, M., Aghababa, M.P., Solouk, V., 2019. Heuristic nonlinear regression strategy for detecting phishing websites. Soft Comput. 23 (12), 4315–4327. https://doi.org/10.1007/s00500-018-3084-2.

Bell, S., Komisarczuk, P., 2020. An analysis of phishing blacklists: Google safe browsing, OpenPhish, and PhishTank. In: Proc. of the ACSW 2020, 2.

Bozkir, A.S., Aydos, M., 2020. LogoSENSE: a companion HOG based logo detection scheme for phishing web page and e-mail brand recognition. Comput. Secur. 95, 101855. https://doi.org/10.1016/j.cose.2020.101855.

Chai, Y.D., Zhou, Y.H., Li, W.F., Jiang, Y.C., 2022. An explainable multi-modal hierarchical attention model for developing phishing threat intelligence. IEEE Trans. Dependable Secure Comput. 19 (2), 790–803. https://doi.org/10.1109/TDSC.2021.3119323.

Chiew, K.L., Yong, K.S.C., Tan, C.L., 2018. A survey of phishing attacks: their types, vectors and technical approaches. Expert Syst. Appl. 106, 1–20. https://doi.org/10.1016/j.eswa.2018.03.050.

Chiew, K.L., Tan, C.L., Wong, K.S., Yong, K.S.C., Tiong, W.K., 2019. A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. Inf. Sci. 484, 153–166. https://doi.org/10.1016/j.ins.2019.01.064.

China Network Security Technical Challenge, 2017. Malicious webpage analysis exercise dataset [Online]. https://www.heywhale.com/mw/project/5c8287121e7104002b3770df.

Ding, Y., Luktarhan, N., Li, K., Slamu, W., 2019. A keyword-based combination approach for detecting phishing webpages. Comput. Secur. 84, 256–275. https://doi.org/10.1016/j.cose.2019.03.018.

Google, 2021. Safe browsing API. [Online]. https://developers.google.cn/safe-browsing/v4?hl=en.

Gupta, B.B., Jain, A.K., 2020. Phishing attack detection using a search engine and heuristics-based technique. J. Inf. Technol. Res. 13 (2), 94–109. https://doi.org/10.4018/JITR.2020040106.

Haruta, S., Asahina, H., Sasase, I., 2017. Visual similarity-based phishing detection scheme using image and CSS with target website finder. In: Proc. of the GLOBECOM 2017, pp. 1–6.

He, X.L., Gong, Q.Y., Chen, Y., Zhang, Y., Wang, X., Fu, X.M., 2021. DatingSec: detecting malicious accounts in dating apps using a content-based attention network. IEEE Trans. Dependable Secure Comput. 18 (5), 2193–2208. https://doi.org/10.1109/TDSC.2021.3068307.

Jai, A.K., Gupta, B.B., 2017. Phishing detection: analysis of visual similarity based approaches. Secur. Commun. Netw. 2017, 5421046. https://doi.org/10.1155/2017/5421046.

Jain, A.K., Gupta, B.B., 2015. PHISH-SAFE: URL features-based phishing detection system using machine learning. In: Proc. of the CIS 2015, pp. 467–474.

Jain, A.K., Gupta, B.B., 2016. Comparative analysis of features based machine learning approaches for phishing detection. In: Proc. of the INDIACom 2016, pp. 2125–2130.

Jain, A.K., Gupta, B.B., 2022. A survey of phishing attack techniques, defence mechanisms and open research challenges. Enterp. Inf. Syst. 16 (4), 527–565. https://doi.org/10.1080/17517575.2021.1896786.

Jain, A.K., Gupta, B.B., Kaur, K., Bhutani, P., Alhalabi, W., Almomani, A., 2022. A content and URL analysis-based efficient approach to detect smishing SMS in intelligent systems. Int. J. Intell. Syst. 37 (12), 11117–11141. https://doi.org/10.1002/int.23035.

Khonji, M., Iraqi, Y., Jones, A., 2013. Phishing detection: a literature survey. IEEE Commun. Surv. Tutor. 15 (4), 2091–2121. https://doi.org/10.1109/SURV.2013.032213.00009.

Kuo, C.L., Kuruoglu, E.E., Chan, W.K.V., 2022. Neural network structure optimization by simulated annealing. Entropy 24 (3), 348. https://doi.org/10.3390/e24030348.

Li, Q., Cheng, M.Y., Wang, J.F., Sun, B.W., 2022. LSTM based phishing detection for big email data. IEEE Trans. Big Data 8 (1), 278–288. https://doi.org/10.1109/TBDATA.2020.2978915.

Liang, Y.J., Wang, Q.S., Xiong, K., Zheng, X.L., Yu, Z.W., Zeng, D., 2022. Robust detection of malicious URLs with self-paced wide & deep learning. IEEE Trans. Dependable Secure Comput. 19 (2), 717–730. https://doi.org/10.1109/TDSC.2021.3121388.

Lin, T., Capecci, D.E., Ellis, D.M., Rocha, H., Dommaraju, S., Oliveira, D.A.S., Ebner, N.C., 2019. Susceptibility to spear-phishing emails: effects of Internet user demographics and email content. ACM Trans. Comput.-Hum. Interact. 26 (5), 32. https://doi.org/10.1145/3336141.

Marchal, S., Francois, J., State, R., Engel, T., 2014. PhishStorm: detecting phishing with streaming analytics. IEEE Trans. Netw. Serv. Manag. 11 (4), 458–471. https://doi.org/10.1109/TNSM.2014.2377295.

Marchal, S., Armano, G., Grondahl, T., Saari, K., Singh, N., Asokan, N., 2017. Off-the-hook: an efficient and usable client-side phishing prevention application. IEEE Trans. Comput. 66 (10), 1717–1733. https://doi.org/10.1109/TC.2017.2703808.

Microsoft, 2023. Windows SmartScreen [Online]. https://support.microsoft.com/zh-cn/windows/windows-7c7f6a09-cebd-5589-c376-7f505e5bf65a.

Mirian, A., DeBlasio, J., Savage, S., Voelker, G.M., Thomas, K., 2019. Hack for hire: exploring the emerging market for account hijacking. In: Proc. of the WWW 2019, pp. 1279–1289. https://doi.org/10.1145/3308558.3313489.

Mishra, A., Gupta, B.B., Perakovic, D., Penalvo, F.J.G., Hsu, C., 2021. Classification based machine learning for detection of DDoS attack in cloud computing. In: Proc. of the ICCE 2021, pp. 1–4.

Mohammad, R.M., Thabtah, F., McCluskey, L., 2014. Predicting phishing websites based on self-structuring neural network. Neural Comput. Appl. 25 (2), 443–458. https://doi.org/10.1007/s00521-013-1490-z.

Oest, A., Safaei, Y., Doupe, A., Ahn, G., Wardman, B., Tyers, K., 2019. PhishFarm: a scalable framework for measuring the effectiveness of evasion techniques against browser phishing blacklists. In: Proc. of the SP 2019, pp. 1344–1361.

Oest, A., Safaei, Y., Zhang, P., Wardman, B., Tyers, K., Shoshitaishvili, Y., Doupe, A., 2020. PhishTime: continuous longitudinal measurement of the effectiveness of anti-phishing blacklists. In: Proc. of the USENIX Security 2020, pp. 379–396.

Opara, C., Chen, Y.K., Wei, B., 2024. Look before you leap: detecting phishing web pages by exploiting raw URL and HTML characteristics. Expert Syst. Appl. 236, 121183. https://doi.org/10.1016/j.eswa.2023.121183.

OpenDNS, 2022. Phishtank [Online]. https://phishtank.org/.

OpenPhish, 2018. OpenPhish-phishing intelligence [Online]. https://openphish.com/.

Oram, E., Dash, P.B., Naik, B., Nayak, J., Vimal, S., Nataraj, S.K., 2021. Light gradient boosting machine-based phishing webpage detection model using phisher website features of mimic URLs. Pattern Recognit. Lett. 151, 100–106. https://doi.org/10.1016/j.patrec.2021.09.018.

Pham, C., Nguyen, L.A.T., Tran, N.H., Huh, E., Hong, C.S., 2018. Phishing-aware: a neuro-fuzzy approach for anti-phishing on fog networks. IEEE Trans. Netw. Serv. Manag. 15 (3), 1076–1089. https://doi.org/10.1109/TNSM.2018.2831197.

Piantadosi, S.T., 2014. Zipf's word frequency law in natural language: a critical review and future directions. Psychon. Bull. Rev. 21, 1112–1130. https://doi.org/10.3758/s13423-014-0585-6.

Reasonable, 2021. Reasonable anti-phishing (2.1.0) [Online]. https://www.freshdevices.com/25241/details-reasonable-anti-phishing-software.html.

Sahingoz, O.K., Buber, E., Demir, O., Diri, B., 2019. Machine learning based phishing detection from URLs. Expert Syst. Appl. 117, 345–357. https://doi.org/10.1016/j.eswa.2018.09.029.

Silva, C.M.R., Feitosa, E.L., Garcia, V.C., 2020. Heuristic-based strategy for phishing prediction: a survey of URL-based approach. Comput. Secur. 88, 101613. https://doi.org/10.1016/j.cose.2019.101613.

Sundhari, S.S., 2011. A knowledge discovery using decision tree by Gini coefficient. In: Proc. of the ICBEIA 2011, pp. 232–235.

Tan, C.L., 2018. Phishing dataset for machine learning: feature evaluation, Mendeley data (V1) [Online]. https://data.mendeley.com/datasets/h3cgnj8hft/1. https://doi.org/10.17632/h3cgnj8hft.1.

Valecha, R., Mandaokar, P., Rao, H.R., 2022. Phishing email detection using persuasion cues. IEEE Trans. Dependable Secure Comput. 19 (2), 747–756. https://doi.org/10.1109/TDSC.2021.3118931.

Wazirali, R., Ahmad, R., Abu-Ein, A.A.H., 2021. Sustaining accurate detection of phishing URLs using SDN and feature selection approaches. Comput. Netw. 201, 108591. https://doi.org/10.1016/j.comnet.2021.108591.

Wu, J.J., Yuan, Q., Lin, D., You, W., Chen, W.L., Chen, C., Zheng, Z.B., 2022. Who are the phishers? Phishing scam detection on Ethereum via network embedding. IEEE Trans. Syst. Man Cybern. Syst. 52 (2), 1156–1166. https://doi.org/10.1109/TSMC.2020.3016821.

Xiang, G., Hong, J.I., Rose, C.P., Cranor, L.F., 2011. CANTINA+: a feature-rich machine learning framework for detecting phishing web sites. ACM Trans. Inf. Syst. Secur. 14 (2), 21. https://doi.org/10.1145/2019599.2019606.

Xiao, X., Zhang, D.Y., Hu, G.W., Jiang, Y., Xia, S.T., 2020. CNN-MHSA: a Convolutional Neural Network and multi-head self-attention combined approach for detecting phishing websites. Neural Netw. 125, 303–312. https://doi.org/10.1016/j.neunet.2020.02.013.

Zabihimayvan, M., Doran, D., 2019. Fuzzy Rough Set feature selection to enhance phishing attack detection. In: Proc. of the FUZZ-IEEE 2019, pp. 1–6.

Zeng, V., Zhou, X., Baki, S., Verma, R.M., 2020. PhishBench 2.0: a versatile and extendable benchmarking framework for phishing. In: Proc. of the CCS 2020, pp. 2077–2079.

Zhang, P., Oest, A., Cho, H., Sun, Z., Johnson, R.C., Wardman, B., Sarker, S., Kapravelos, A., Bao, T., Wang, R., Shoshitaishvili, Y., Doupe, A., Ahn, G., 2022. CrawlPhish: large-scale analysis of client-side cloaking techniques in phishing. IEEE Secur. Priv. 20 (2), 10–21. https://doi.org/10.1109/MSEC.2021.3129992.

Zhu, E.Z., Ju, Y.Y., Chen, Z.L., Liu, F., Fang, X.Y., 2020. DTOF-ANN: an artificial neural network phishing detection model based on decision tree and optimal features. Appl. Soft Comput. 95, 106505. https://doi.org/10.1016/j.asoc.2020.106505.

Zhu, E.Z., Chen, Z.L., Cui, J., Zhong, H., 2022. MOE/RF: a novel phishing detection model based on revised multi-objective evolution optimization algorithm and random forest. IEEE Trans. Netw. Serv. Manag. 19 (4), 4461–4478. https://doi.org/10.1109/TNSM.2022.3162885.

Zhuang, W.W., Ye, Y.F., Chen, Y., Li, T., 2012. Ensemble clustering for Internet security applications. IEEE Trans. Syst. Man Cybern., Part C, Appl. Rev. 42 (6), 1784–1796. https://doi.org/10.1109/TSMCC.2012.2222025.

Zouina, M., Outtaj, B., 2017. A novel lightweight URL phishing detection system using SVM and similarity index. Hum.-Cent. Comput. Inf. Sci. 7, 17. https://doi.org/10.1186/s13673-017-0098-1.

Erzhou Zhu was born in Anhui, China, in 1981. He received the Ph.D. degree from the Shanghai Jiao Tong University in 2012. He is currently an Associate Professor and the dean of department of Information Security, Anhui University. He has over 20 scientific publications in journals, including the IEEE Transactions on Network and Service Management, the Computers & Security, the Cognitive Computation, the Journal of Systems Architecture, the Applied Soft Computing, the Neurocomputing, the Computers & Electrical Engineering, the Journal of Information Science and Engineering, and the Journal of Software (in Chinese). His current research interests include data mining, machine learning, and IoT security.

Kang Cheng is currently a Research Student with the School of Computer Science and Technology, Anhui University, Hefei, China. Her research interests include IoT security and machine learning.

Zhizheng Zhang is currently a Research Student with the School of Computer Science and Technology, Anhui University, Hefei, China. Her research interests include IoT security and machine learning.

Huabin Wang was born in Anhui, China, in 1983. He received the Ph.D. degree in computer application technology from Anhui University, Hefei, China, in 2011. He is currently an Associate Professor and the dean of department of Computer Science and Technology, Anhui University. He has over 10 scientific publications in journals, including the IEEE Transactions on Network Science and Engineering, the Cognitive Computation. His research interests include machine learning, pattern recognition and signal processing.