

DPMLF: A Novel Phishing Detection Model Integrating URL Character and HTML Semantic Deep Features

Zhu-Juan Ma¹, Xiao-Han Liu², Li-Hui Meng², and Er-Zhou Zhu^{2*}

¹ School of Big Data and Artificial Intelligence, Anhui Xinhua University, Hefei 230088, China
zjmsjtu16@163.com

² School of Computer Science and Technology, Anhui University, Hefei 230601, China
u24301305@163.com, e22201138@stu.ahu.edu.cn, ezzhu@ahu.edu.cn

Received 2 February 2025; Revised 18 July 2025; Accepted 4 August 2025

Abstract. Deep learning (DL) methods have been extensively applied in phishing detection, owing to their powerful feature-learning and classification capabilities. Nevertheless, generating deep feature vectors that can precisely characterize phishing behaviors is essential to effectively recognize and detect phishing attacks within the cyber realm. This study proposes DL phishing detection with multilevel features (DPMLF), a novel model crafted to boost phishing detection performance by integrating uniform resource locator (URL) character-level and HyperText Markup Language (HTML) word-level semantic features. DPMLF uses character embedding along with parallel convolutional kernels of diverse sizes to extract local URL features. HTML text adopts word-level embedding and stacked convolutional layers to capture both local and long-range dependencies. Subsequently, a fully connected layer merges these multilevel features into a fine-grained vector for classification. Moreover, DPMLF incorporates an incremental learning mechanism, enabling it to continuously update its knowledge base with new data. When evaluated on three public datasets sourced from GitHub, Kaggle, and Zenodo, DPMLF attained an average accuracy of 0.9900, precision of 0.9901, recall of 0.9899, and an F_1 -score of 0.9900.

Keywords: phishing detection, feature fusion, deep learning, cyberspace security, incremental learning

1 Introduction

Phishing is a form of network attack grounded in social engineering. In this scheme, attackers disguise phishing websites as resembling the “legitimate websites” of actual organizations. They entice users to click on links or enter login information, thereby fulfilling their objectives of launching network assaults, stealing sensitive data, or seizing the control of target platforms [1]. Phishing is a significant threat to cyberspace security. Simultaneously, phishing incidences are on the rise. According to reports by the Anti-Phishing Working Group (APWG) [2], a total number of 3,763,576 phishing attacks were detected in 2024. Their reports further indicated that, in the third quarter of 2024, phishing attacks targeting social media platforms accounted for 30.5% of all attacks. In the face of escalating phishing threats, an urgent need exists for effective detection and prevention strategies that safeguard user networks and information security.

In addition to educating users to identify phishing attempts quickly, various automated methods have been proposed to mitigate the threat of phishing attacks. These methods include blacklisting, heuristic-based detection, data mining, visual similarity checking, and machine learning (ML)-based classification [3]. The blacklist approach, which relies on a predefined set of uniform resource locators (URLs), can rapidly and accurately identify known phishing activities. However, it encounters limitations when handling newly emerging phishing URLs. Visual similarity methods, which compare the visual characteristics of websites, can effectively identify potential phishing instances but are computationally demanding [4]. ML methods detect malicious websites by analyzing features such as URLs, static HyperText Markup Language (HTML) content, page document object model (DOM), and screenshots. These methods construct optimal feature vectors to train classification models and achieve high phishing detection accuracy. Nevertheless, their performance is highly reliant on manually crafted features derived from expert knowledge, which makes it difficult to address novel and evolving phishing behaviors [5]. Deep learning (DL) methods, which are not dependent on expert a priori knowledge, can automatically

* Corresponding Author

and precisely learn the relevant features of network behaviors. Consequently, they have been extensively applied to phishing detection models [6].

For DL-based phishing detection models, generating feature vectors that accurately capture web behavior remains a formidable challenge. This is owing to the difficulties in fusing the semantic information of URL characters and HTML content across multiple levels of fine-grained granularity. The semistructured nature of webpages complicates automatic feature extraction. Current studies often focus on a single aspect, such as URL or HTML content, resulting in an incomplete understanding of webpages and diminished detection efficacy [7]. Moreover, phishing websites often maintain semantic resemblance to legitimate branded websites, including similarities in URLs, titles, and HTML content, to deceive users. Existing detection methods predominantly extract features heuristically, and frequently overlook the fact that webpages comprise multiple functional modules [8]. Owing to update delays in the training dataset, the DL-based phishing detection method has a major limitation in recognizing websites with novel or unfamiliar patterns (i.e., zero-day phishing websites).

This study proposes DL phishing detection with multilevel features (DPMLF), a novel DL model for phishing website detection. DPMLF integrates multilevel fine-grained semantic information from URL characters and HTML content. The model extracts features from various components of a webpage and fuses them, and then uses an integrated vector to enhance detection performance. HTML content provides semantic and structural information, whereas the URL offers address-related information. The model capitalizes on complementary features at multiple levels to improve the recognition of phishing websites by integrating these two sources. The main contributions of this study are as follows:

- (1) *Fine-grained URL character local features extraction using a multiscale convolutional strategy.* A convolutional neural network (CNN) equipped with differently sized convolutional kernels was employed to process the URL strings in parallel. This approach endows the DL model with the ability to grasp the features of different granularities from the input data.
- (2) *Word-level HTML text feature extraction using a cascaded densely-connected CNN architecture.* This architecture stacks multiple convolutional layers to capture local features at different scales. This architecture can effectively capture long-range text information by establishing multilayer cascaded dense connections among the CNN blocks, thereby extracting deeper and finer-grained semantic features from HTML documents.
- (3) *URL character-level and HTML word-level semantic features fusion mechanism using a fully-connected layer.* The final constructed DPMLF model can comprehensively characterize the features of phishing websites by fusing the URL character and HTML text information.
- (4) *Fine-tuning the DPMLF model using an incremental learning mechanism.* An incremental learning mechanism is used to continuously update the DPMLF model using new phishing data to adapt to new phishing threat patterns.

The remainder of this paper is organized as follows: Section 2 discusses related work. Section 3 presents an overview and the implementation of the proposed DPMLF model. In Section 4, experimental results are presented and discussed. Finally, Section 5 concludes the paper and outlines the directions for future studies.

2 Related Work

In addition to educating users about quickly identifying phishing attempts, perhaps the most straightforward approach is to build anti-phishing blacklists. Blacklists are a vital defense mechanism utilized by modern web browsers to prevent large-scale phishing attacks. For instance, mainstream browsers such as Google Chrome, Mozilla Firefox, and Apple Safari are integrated with Google Safe Browsing [9], a security service provided by Google that includes a regularly updated blacklist. As new domains constantly emerge, blacklists must be updated at an ever-increasing pace to keep pace with evolving threat landscapes. Oest et al. [10] proposed the phishing-time framework, which continuously assesses the effectiveness of anti-phishing blacklists. Skula et al. [11] performed an in-depth analysis of the phishing data from 2013 to 2022. Their findings revealed that approximately 22% of the phishing domains reappeared. This discovery validated the effectiveness of blacklist-based detection methods. A blacklist was created by collecting numerous reported phishing URLs. This method does not require access to the web content; therefore, it is highly efficient for phishing. However, updating a blacklist requires time and the latest URLs [12].

Heuristic-based detection methods classify phishing attacks by extracting a set of features from URLs, webpage layouts, HTML, JavaScript, and other website components. Based on an assessment of the pertinence of

contemporary phishing activities, Silva et al. [13] employed 12 features from keywords and patterns in URLs for phishing detection. Based on the observation that phishing websites are short-lived and created for a specific purpose, Shukla et al. [14] extracted 16 new HTTP header features and used them to train an ML-based phishing detection model. Rao et al. [15] introduced a dual-support vector machine methodology. This approach involved comparing the login and homepage of a website. Malicious phishing websites can be effectively identified by meticulously analyzing hyperlinks and URL characteristics. The key to heuristic-based phishing detection is in leveraging prior knowledge and experience-driven rules. As phishing techniques evolve, continuous refinement of these heuristic rules is necessary to maintain detection effectiveness. The PDHF [3] model was constructed to address the issue of phishing attackers bypassing constructed filters after learning the heuristic rules used for detection by combining optimal artificial and automatic deep features. The heuristic method solves the time lag in phishing detection. However, it relies on building heuristic rules and incurs a high false positive rate, which may reduce users' trust in the phishing detection system [16].

Data mining-based phishing detection typically involves two steps: feature extraction and classification. Li et al. [17] improved the performance in identifying phishing addresses by mining the temporal relationship of historical transaction records among different nodes in the Ethereum transaction network. Liu et al. [18] developed the PGDetector model to identify risky accounts from large-scale Ethereum transaction data. Their approach combined data-mining techniques with genetic-algorithm optimization. Luo et al. [19] proposed a phishing-account detection model grounded in network embeddings that specifically targeted transaction-security concerns on the Ethereum platform. This model captures and constructs transaction networks by incorporating supplementary information. Subsequently, classifiers such as lightGBM and XGBoost were employed to classify the phishing accounts. Data mining approaches consider the detection of phishing attacks as a document classification problem or clustering problem, and data models are constructed based on ML and clustering algorithms [20]. However, the effectiveness of these approaches often depends on the quality and representativeness of the data used for training and the appropriateness of the selected data-mining algorithms [21]. For instance, clustering algorithms, such as k-means and k-nearest neighbors, partition target datasets in an unsupervised manner. Improper setting of the number of clusters and initial clustering centers can severely degrade the quality of the clustering results [22].

Visual similarity detection methods assume that most phishing websites are visually similar to the legitimate websites that they imitate. VisualPhishNet [23] compared the visual features of webpages with those of known legitimate websites to identify phishing websites by capturing screenshots of webpages and analyzing their visual features using a DL model. Wang et al. [24] first used a DL model to recognize website logos. The recognized results and overall webpages are then combined to enhance the performance of the phishing webpage detection. Tan et al. [25] proposed a hybrid phishing detection method that integrates both visual and textual identity information to classify webpages. Wen et al. [26] developed an expansion method to identify suspicious websites by analyzing existing phishing sites. This method segments screenshots to accurately extract brand logos. Visual similarity-checking methods are effective in detecting phishing when a legitimate counterpart of a phishing site is already registered in the detection system. However, delayed updates of legitimate websites in the detection system limit its ability to identify websites with unknown brand representations (such as brand logos, UI components, and website screenshots), rendering it ineffective against zero-day phishing webpages [27].

Owing to its strong learning and classification capabilities, ML has been extensively applied to several phishing detection models. Bahaghighat et al. [28] proposed an enhanced ML prediction model to improve the efficiency of anti-phishing technology. This model evaluated eight scenarios using six algorithms after feature selection and data balancing. Silva et al. [29] proposed Piracema.io, a rule-based phishing detection model that utilizes decision trees to classify phishing behaviors into five classes. The MOE/RF model [30] combines multiobjective evolutionary optimization and random forests to identify the optimal feature vector for maximizing both accuracy and recall in phishing detection. ML-based phishing models consider phishing detection to be a classification problem. Researchers often extract numerous features to enhance phishing detection accuracy [31]. Nevertheless, phishing attackers gradually adapt to several of these features over time. Therefore, maintaining the long-term effectiveness of phishing detection models based on artificial features is challenging [32].

As a subset of ML, DL has gained widespread application in several fields. It uses deep neural networks (DNNs) to extract deep features directly from datasets, thereby avoiding cumbersome feature engineering processes. Wen et al. [33] proposed a hybrid DNN model to detect phishing accounts on Ethereum. Hussain et al. [6] developed a CNN-fusion model that extracted character sequence features using convolutional kernels of different sizes to identify malicious URLs. He et al. [34] developed a tiny-BERT stacking model. Tiny-BERT extracts and learns semantic and long-range features from URLs to mitigate phishing risks. Liu et al. [8] analyzed the word, phrase, and code structure features from the source code of webpages to accurately capture the character-

istics of complex phishing websites. Because they are not dependent on experts' a priori knowledge, DL-based phishing detection methods can automatically and precisely learn the relevant features of network behavior [35]. However, the quality of automatic DL features highly depends on the scale of the datasets to be learned [36].

For DL-based phishing detection models, generating deep feature vectors that can accurately capture web behavior remains a formidable challenge: (1) The semi-structured nature of webpages complicates automatic feature extraction. Numerous existing detection methods generate phishing features predominantly heuristically and frequently overlook the fact that webpages comprise multiple functional modules [8]. (2) Phishing websites often maintain a semantic resemblance to legitimate branded websites, including similarities in URLs, titles, and HTML content, to deceive users. Because of the difficulties in fusing the semantic information of URL characters and HTML content across multiple levels of fine-grained granularity, numerous current studies often focus on a single aspect, such as the URL or HTML content, resulting in an incomplete understanding of webpages and diminished detection efficacy [7].

However, because of update delays in the training dataset, the DL-based phishing detection method has a major limitation in recognizing websites with novel or unfamiliar patterns (i.e., zero-day phishing websites). The performance of these methods is highly dependent on the scale of the dataset to be learned. If a website with a new pattern is not included in the dataset used to train the model, the detection effect of the model on the website will be poor. Although some recent phishing detection models, such as the generative pretrained transformer (GPT) and Google Images, access latency and payment costs associated with third-party services severely restrict their widespread adoption [27].

The DPMLF phishing detection model proposed in this study is a DL method; therefore, it inherits the advantages of DL methods. Moreover, it effectively overcomes these shortcomings. The DPMLF model extracts features from various components of a webpage and fuses them, and subsequently uses an integrated vector to enhance the detection performance. HTML content provides semantic and structural information, whereas URL offers address-related information. The model capitalizes on complementary features at multiple levels to improve the recognition of phishing websites by integrating these two sources. Furthermore, to adapt to new phishing threat patterns, an incremental learning mechanism is used to continuously update the DPMLF model using new phishing data.

3 Proposed DPMLF

The DPMLF model is a composite DNN designed to learn features directly from raw URL characteristics and HTML content for phishing classification.

3.1 Problem Definition

Consider a dataset $P = \{(u_1, h_1, y_1), (u_2, h_2, y_2), \dots, (u_n, h_n, y_n)\}$ containing n webpages, where $P_i = (u_i, h_i, y_i)$ represents a webpage sample and u_i , h_i , and y_i denote the URL, HTML content, and label of P_i respectively. Specifically, $y_i = 0$ indicates a legitimate webpage and $y_i = 1$ indicates a phishing webpage. The DPMLF model transforms phishing detection into a binary classification problem. Its primary objective is to train a predictive function that automatically generates a representation for each webpage P_i , denoted as $P_i \rightarrow X_i$, where X_i denotes the feature matrix of P_i . Subsequently, the extracted feature matrices X_i are used to train a discriminative model $f: X \rightarrow Y$, which is then employed to classify unseen webpages.

3.2 Overview of DPMLF

As illustrated in Fig. 1, the DPMLF workflow primarily comprises three stages: embedding, deep feature extraction, and website classification.

In the embedding stage, the URL text and HTML content are first refined into character-level and word-embedding representations, respectively. Thereafter, they are mapped to low-dimensional continuous vector representations to effectively capture character and semantic features, that is, to achieve semantic embedding. Finally, batch normalization (BN) was performed to accelerate the convergence of the model.

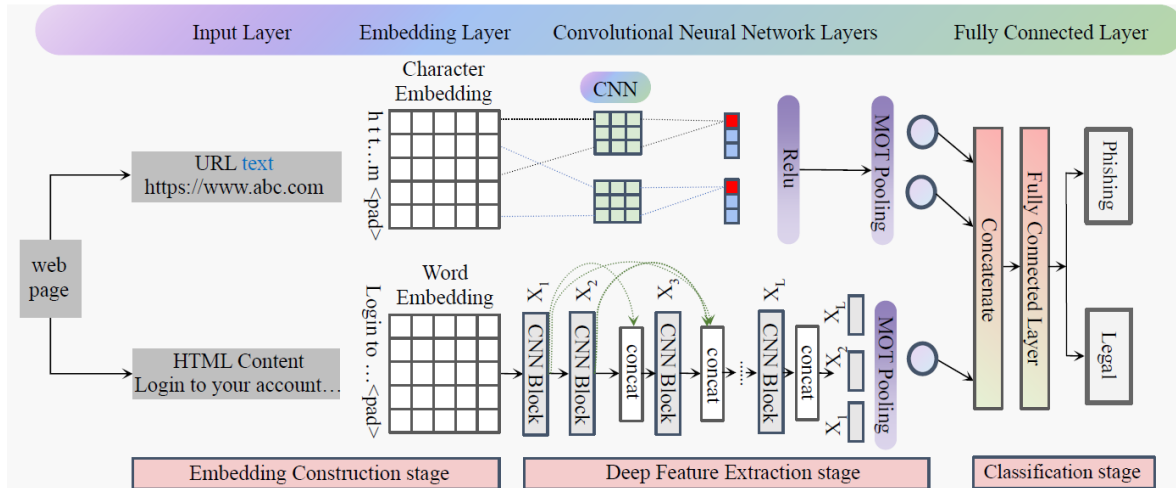


Fig. 1. Architecture of the DPMLF model

The deep feature extraction stage uses embeddings as inputs for convolutional operations that extract continuous character sequence features from URLs, identifying local character relationships. Each convolutional layer is followed by a rectified linear unit (ReLU) activation function, and max-over-time pooling (MOT pooling) retains key features. For the HTML content, a densely connected CNN cascaded structure was used. Multiple convolutional layers with dense connections between the blocks extract local and long-range text features. The final output of each block is generated through MOT pooling.

In the deep-feature extraction stage, text representations generated by the embedding layer were used as inputs. Convolution operations are first employed to extract the features of continuous character sequences of different lengths in the URL to understand the local relationships between URL characters fully. Accordingly, the ReLU activation function and MOT pooling operations were successively applied to retain the key features at each time step.

Simultaneously, a CNN with cascaded dense connections was used to extract features from the HTML content. Local features at different scales and long-range text information can be extracted by stacking multiple convolutional layers and establishing dense connections between the convolutional blocks. On this basis, the convolution results of each module were recorded, and HTML text features were generated through the MOT pooling operation.

In the classification stage, the extracted URL and HTML features are first concatenated to form a feature vector that comprehensively characterizes the target websites. Thereafter, the generated feature vector was fed into the fully connected layer to determine whether the target website contained malicious behavior.

3.3 Character and Word Embedding

Phishing URLs often employ obfuscation techniques, such as the insertion of special and random characters, making it challenging for traditional word embedding methods, such as Word2Vec [37] and GloVe [38], to precisely capture subtle variations between characters. Consequently, this study introduces character- and word-level embedding strategies for URL strings and HTML documents. The model can fully capture the textual characteristics of a target webpage by combining the URL and HTML features.

In the embedding stage, the URLs are tokenized at the character level and adjusted to a uniform length through truncation and padding using the <PAD> token. The HTML content was tokenized at the word level, with punctuation marks treated as separate tokens, and the same truncation and padding strategies were applied. Character-level vocabulary is generated from unique characters, including letters, digits, and special characters. HTML documents were divided into words to form a word-level corpus, and the original words were replaced with numerical indices to create one-dimensional (1D) numerical vectors. The effectiveness of phishing website detection can be significantly improved by identifying common sensitive terms, such as “login” and “password,” in the HTML content of webpages.

The embedding layer automatically converts the tokenized characters (from URLs) and words (from HTML

content) into feature vectors. These vectors contain the semantic relationships between URL characters and HTML content, which are essential for the subsequent stages of the model.

Based on the above methods, each URL is represented by a character embedding matrix $U \in R^{n \times d}$, where n denotes the number of characters in the URL and d denotes the embedding dimension. For each HTML document, a word-embedding matrix $H \in R^{m \times d}$ was constructed, where m denotes the number of words in the document and d denotes the embedding dimension.

3.4 Multigranularity Feature Extraction from URLs

We employed CNNs with different kernel sizes to extract semantic information at different granularities from URL strings. The convolutional stage utilizes a weight-sharing mechanism to reduce the number of parameters and the model complexity. Meanwhile, the pooling operation retains key features to reduce data volume. As illustrated in Fig. 2, during 1D convolution, each local window is applied to the character embedding sequence, and a new vector is generated by linear combinations of weights.

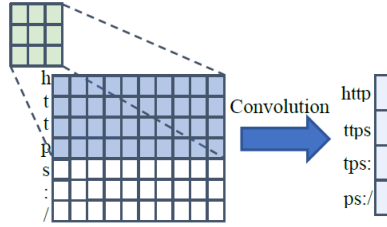


Fig. 2. 1D convolution

After the embedding layer, the URL characteristics are transformed into a sequence of vectors, as shown in Equation (1):

$$U = [e_1, e_2, \dots, e_n], \quad (1)$$

where $U \in R^{n \times d}$ denotes the character embedding matrix of the target URL, e_i denotes the embedding vector of the i th character, n denotes the length of the target URL, and d denotes the character embedding dimension.

As indicated in Equation (2), convolutional kernels are used to extract feature information from each character window in the string of the target URL, and feature maps are generated.

$$c_i = f_1(w \otimes e_{i:i+h-1}) + b. \quad (2)$$

In Equation (2), c_i denotes the new feature generated from the character window $e_{i:i+h-1}$; $w \in R^{k \times h}$ denotes the weight matrix with k channels and kernel size h ; b denotes the bias term; \otimes represents the convolution operation; and f_1 denotes the ReLU activation function.

The convolution operation is then applied to the URL sequence, and new feature vectors c are generated as follows:

$$c = [c_1, c_2, \dots, c_{n-h+1}]. \quad (3)$$

Finally, as shown in Equation (4), the MOT pooling operation is applied to retain the key features at each time step:

$$f_{URL} = MOTPooling(c). \quad (4)$$

3.5 HTML Multilevel Semantic Feature Extraction

HTML documents are semistructured texts that often require parsing before classification. Even after preprocessing during the embedding phase, noise and incoherent content in HTML may still affect semantic understanding and final classification performance. Traditional CNNs are adept at extracting local features but struggle to capture long-range semantic information in extensive HTML texts. In contrast, recurrent neural networks (RNNs) are effective for grasping global features but struggle with feature fragmentation owing to HTML noise.

Inspired by the work in [39], this study proposes a multilayer convolutional architecture with dense interblock connections to capture different-scale local features and long-distance textual information. MOT pooling was used to extract the HTML features by recording the outputs of each convolutional module. This design enhances the integration of local and global semantic information, thereby significantly improving the classification accuracy.

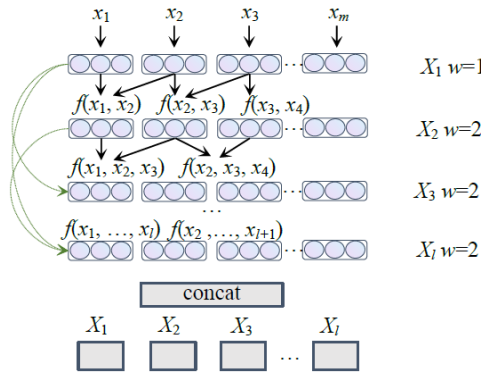


Fig. 3. Cascaded dense CNN

As shown in Fig. 3, the embedding layer transforms HTML text into a sequence of vectors:

$$H = [x_1, x_2, \dots, x_m], \quad (5)$$

where $H \in R^{m \times d}$ represents the word-embedding matrix of the HTML text of the target website, x_i denotes the embedding vector of the i th HTML word, m denotes the length of the HTML content-token sequence, and d denotes the dimension of each word embedding.

After convolution, the output of each intermediate layer can be approximately represented as follows:

$$X_l = [x_l^1, x_l^2, \dots, x_l^m], \quad (6)$$

where $X_l \in R^{m \times k}$, l denotes the index of each layer, and k denotes the feature dimension.

As shown in Equation (7), the model generates the current layer's feature maps by densely connecting the former layer output X_1, X_2, \dots, X_{l-1} .

$$X_l = f(w_l, [X_1, X_2, \dots, X_{l-1}]), \quad (7)$$

where $[X_1, X_2, \dots, X_{l-1}]$ are the feature maps produced by layers 1 to $l-1$, $X_l \in R^{m \times k}$, $w_l \in R^{(l-1) \times k \times w \times k}$ is the weight matrix, with $(l-1) \times k$ being the sum of the feature map channels from the previous layers, w denotes the kernel size, k denotes the number of output channels, and the composite function f comprises convolution, BN, and the ReLU activation function.

Finally, as specified in Equation (8), the final feature representation of the target HTML content is formulated

by concatenating the features from each layer and applying the MOT Pooling operation:

$$f_{HTML} = MOTPooling ([X_1, X_2, \dots, X_l]) . \quad (8)$$

3.6 Feature Fusion in the Fully Connected Layers and Phishing Classification

In the DPMLF model, as shown in Equation (9), the characteristic features of the URL and the content features of the HTML are concatenated and fed into two fully connected layers for in-depth semantic fusion:

$$f_{TOTAL} = Concatenate (f_{URL}, f_{HTML}), \quad (9)$$

where f_{TOTAL} denotes the comprehensive feature representation of the target webpage, f_{URL} and f_{HTML} denote the URL and HTML content features generated by Equations (4) and (8), respectively.

As shown in Equation (10), the generated feature f_{TOTAL} is used to produce the final classification prediction Y_{pred} :

$$Y_{pred} = sigmod (W \times f_{TOTAL} + b) , \quad (10)$$

where W denotes the weight matrix, b denotes the bias term, and Y_{pred} represents the predicted result.

3.7 Incremental Learning Training Process

Given the rapid evolution of phishing attacks, classical ML models face significant challenges in adapting to new scenarios, owing to the need for periodic retraining. Through the continuous integration of new data and meticulous fine-tuning of the present model, incremental learning can rapidly and accurately identify newly emerging phishing threats [40]. As shown in Fig. 4, the incremental learning training process comprises two distinct stages. In the initial training stage, the model is trained using a subset of dataset records. This allows the model to comprehensively understand the underlying patterns. In the incremental-training update stage, new data are continuously incorporated and the model is adjusted to adapt to newly emerging threats.

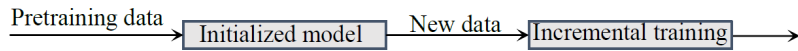


Fig. 4. Incremental learning process of DPMLF

4 Experiment and Analysis

The experiments described in this section were conducted on a computer equipped with an Intel Xeon CPU (4210R2.40 GHz), an NVIDIA GeForce RTX 3070 GPU (8 GB), DDR4 RAM (24 GB), and an Ubuntu operating system (18.04.6 LTS). The programming environment comprised Python programming language (3.8.18) leveraging ML libraries, such as PyTorch 1.12.1 and scikit-learn 1.3.2.

The primary goals of the experiments were to (1) obtain the optimal configuration of the DPMLF model through hyperparameter optimization, (2) validate and assess the performance of the DPMLF model using standard datasets, and (3) verify the overall performance of the DPMLF model by comparing it with other existing models.

As listed in Table 1, three publicly accessible datasets were employed to train and test the proposed DPMLF model. Dataset 1 was sourced from Github [41], Dataset 2 was obtained from Kaggle [7], and Dataset 3, a comprehensive and multivariate dataset, was retrieved from Zenodo [42]. Subsequently, each dataset was partitioned into a training set and a test set in an 8:2 ratio.

Table 1. Description of datasets

Datasets	Web tags	Original sample size	Sampling	Total sample size	Data source
Dataset1	Phishing	21303	21303	46103	PhishTank
	Legal	24800	24800		Alexa
Dataset2	Phishing	22687	22687	45373	PhishTank
	Legal	22686	22686		Alexa
Dataset3	Phishing	40000	32000	64000	Synthesize
	Legal	400000	32000		Synthesize

4.1 Evaluation Metrics

In this study, phishing detection is framed as a binary classification problem, where phishing websites are taken as positive samples and legitimate websites are taken as negative samples. The performance of the DPMLF 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 F₁-score, can be defined to comprehensively evaluate phishing detection performance.

4.2 Determination of URL Length

A vector that is excessively short may lead to a loss of critical information, whereas an excessively long vector can introduce irrelevant padding, thus affecting the model's assessment of URL legitimacy [43]. The impact of URL length variations on the detection accuracy of DPMLF was evaluated using three datasets. This experiment only performed an accuracy test of the model based on URLs, without involving HTML.

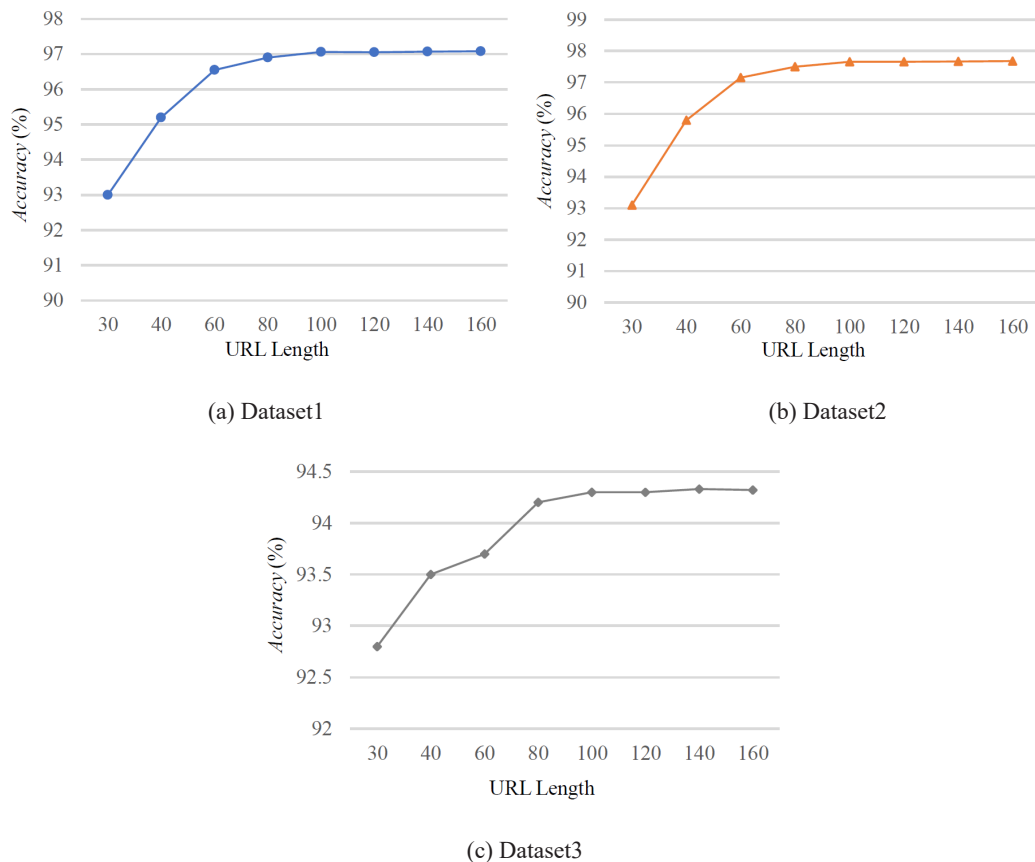
**Fig. 5.** Impact of URL length on accuracy

Fig. 5 shows the variation trend of the model's detection accuracy with an increase in URL length across the three datasets. As observed from the experimental data shown in the figure, the detection accuracy of the model improved significantly during the initial growth stage of the URL. The URLs of the most legitimate websites are within 100 characters in length. Each increment in URL length, up to 100, captured a significant number of complete URLs from the three datasets. However, when the URL length exceeded 100, the improvement in the detection accuracy of the model was no longer evident. As shown in Fig. 5, the average detection accuracy peaks at 100 characters. In particular, because this may cause the model to overfit, further increases in length may result in a decline in accuracy. Therefore, setting the URL length to 100 can maximize the detection accuracy while maintaining information integrity.

4.3 Model Training

In the training phase of the DPMLF model, the Adam optimizer was employed to update the parameters at an initial learning rate of 0.001. A weight decay mechanism was implemented in which the weight decay coefficient was set to 0.0001 to mitigate the issue of overfitting. This strategy effectively bolsters the generalizability of the model by leveraging L2 regularization. To verify the training effect of the DPMLF model, we recorded the changes in the accuracy and loss function curves of the DPMLF on three datasets.

Fig. 6 records the change of accuracy and loss function curves of the DPMLF model on Dataset1. As can be seen from Fig. 6(b), the loss function of DPMLF on the testing set drops rapidly within the first five epochs, and reaches its minimum value at the 10th epoch; from Fig. 6(a), it can be seen that the *Accuracy* of the DPMLF model on the testing set can reach 99% at the 5th epoch, indicating that the model has a relatively fast convergence speed.

Fig. 7 records the change of accuracy and loss function curves of the DPMLF model on Dataset2. As shown in Fig. 7(b), the loss function of DPMLF on the training set drops rapidly within the first eight epochs, and becomes stable afterward. The loss function of the testing set reaches the minimum at the 8th epoch, and then starts to rise owing to overfitting. At this point, training can be terminated early to retain the optimal performance (as shown in Fig. 7(a)) of the model.

Fig. 8 records the change of accuracy and loss function curves of the DPMLF model on Dataset3. As shown in Fig. 8(b), the loss function of DPMLF on the testing set reaches the minimum value at the 10th epoch; however, after the 10th epoch, the loss starts to increase owing to the overfitting of the model. At this point, the training can be terminated early to retain the optimal performance (as shown in Fig. 8(a)) of the model.

Based on the above data and analysis, the DPMLF model converged rapidly when trained on the three datasets.

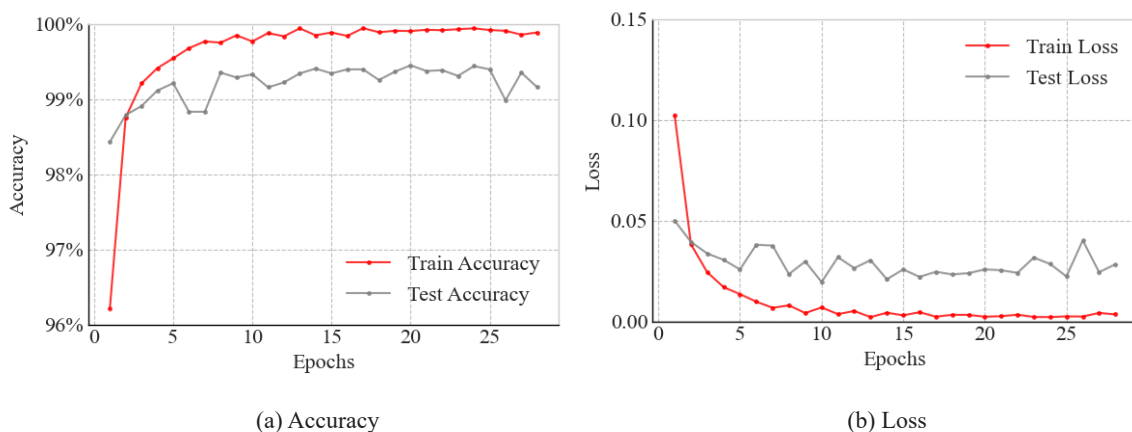


Fig. 6. Accuracy and loss function curves of the DPMLF model on Dataset1

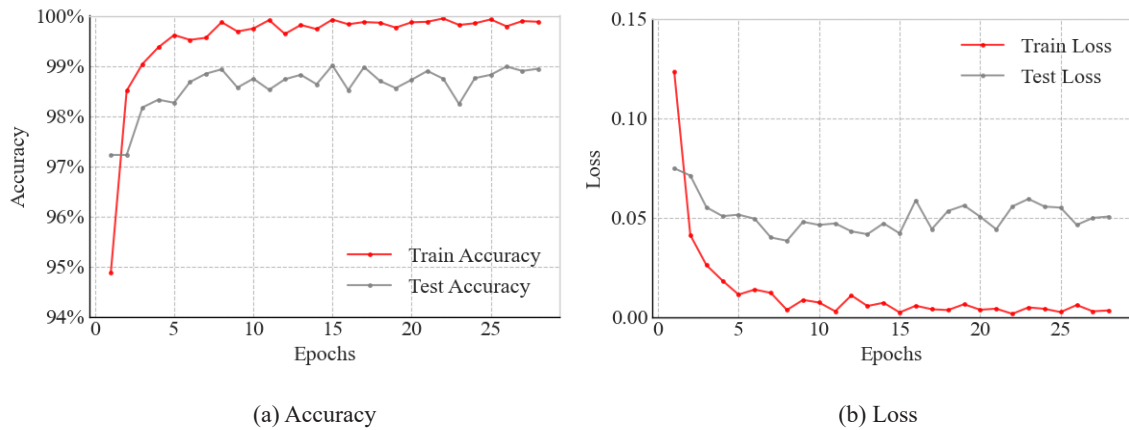


Fig. 7. Accuracy and loss function curves of the DPMLF model on Dataset2

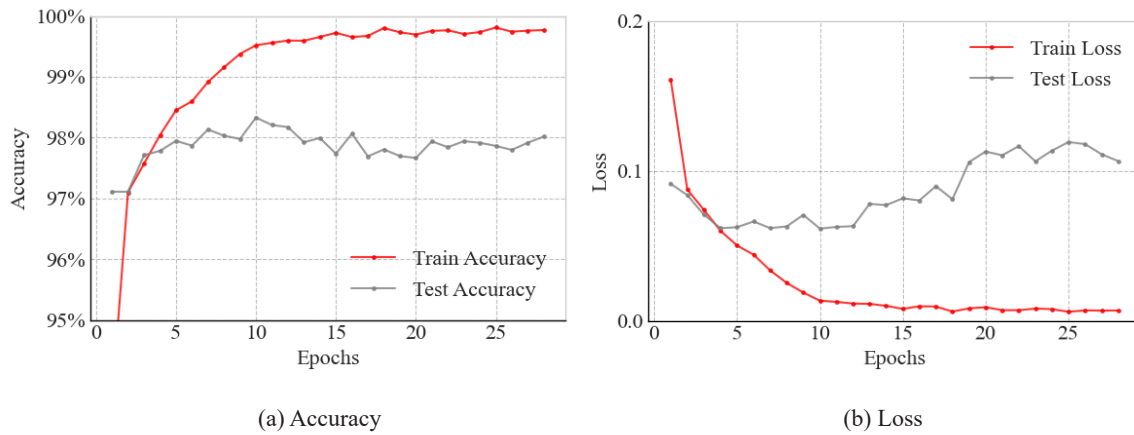


Fig. 8. Accuracy and loss function curves of the DPMLF model on Dataset3

4.4 Determination Hyperparameters of DPMLF

Precisely optimizing the key hyperparameters is essential to precisely optimize the key hyperparameters to enhance the generalization ability and accuracy of the DPMLF model. These hyperparameters include the sizes of the convolution kernels for extracting URL features and the number of cascaded convolutional blocks used to extract the HTML content, which is also known as the network depth. By optimizing these hyperparameters, we can not only accurately evaluate the performance of the model in complex environments but also ensure its effectiveness and reliability in practical applications.

Convolutional Kernel Size. The sizes of the convolution kernels (denoted as n_1 and n_2) are crucial for effectively capturing key information from URLs, specifically for extracting features from continuous character sequence segments. In neural networks, short convolutional kernel sizes have fewer parameters and a higher computational efficiency. However, they may lead to a decrease in model *Accuracy* for reasons such as an extremely poor ability to perceive long-distance dependencies or global patterns and weak nonlinear transformation capabilities that make it difficult to model complex functions. However, as the convolutional kernel size increases, the number of model parameters increases sharply, which directly affects the detection efficiency of the model. Simultaneously, an excessively large convolutional kernel blurs local details, thereby weakening the model's ability to capture and perceive local features.

We adjusted the convolution kernel sizes within the range of 4-15 to obtain the optimal convolution kernel sizes, and the impact of these different sizes on the accuracy of the model was evaluated. In this experiment, a convolutional layer with 64 channels was adopted, the length of the URL sequence was set to a fixed value of

100, and the dimensionality of character embedding was configured as 16. This experiment only conducted an *Accuracy* test of the model based on URLs, without involving HTML.

Experimental validation was performed Dataset1 listed in Table 1, and the comprehensive results are presented in Fig. 9(a). Among all the tested convolution kernel sizes, kernels with lengths of 8 and 12 demonstrated a comprehensive optimal performance. As shown in Fig. 9(a), although large convolutional kernels lead to a certain improvement in accuracy, the sharply increasing number of parameters is bound to consume more memory and computing resources. For instance, in this subfigure, when the convolutional kernel size is 13, the accuracy of the model on the training set is 99.9%, which is only 0.04% higher than that when the convolutional kernel size is 12. Consequently, the model consumed more underlying resources while achieving a greater improvement in accuracy.

Through further experimental validation, such as the experimental results conducted on Dataset2 (Fig. 9(b)) and Dataset3 Fig. 9(c), it was also found that combining the convolution kernel of length 8 with that of length 12 could achieve a comprehensive optimal performance.

Based on the above experimental results, the convolution kernels of different sizes (8 and 12) can extract features with different granularities from the same input URL. The combined use of convolutional kernels with sizes of 8 and 12 can significantly improve the accuracy of the model in detecting phishing URLs. For instance, in subsequent experiments, the model achieved accuracies of 97.06% (Table 2), 97.66% (Table 3), and 94.30% (Table 4) for Dataset1, Dataset2, and Dataset3, respectively.

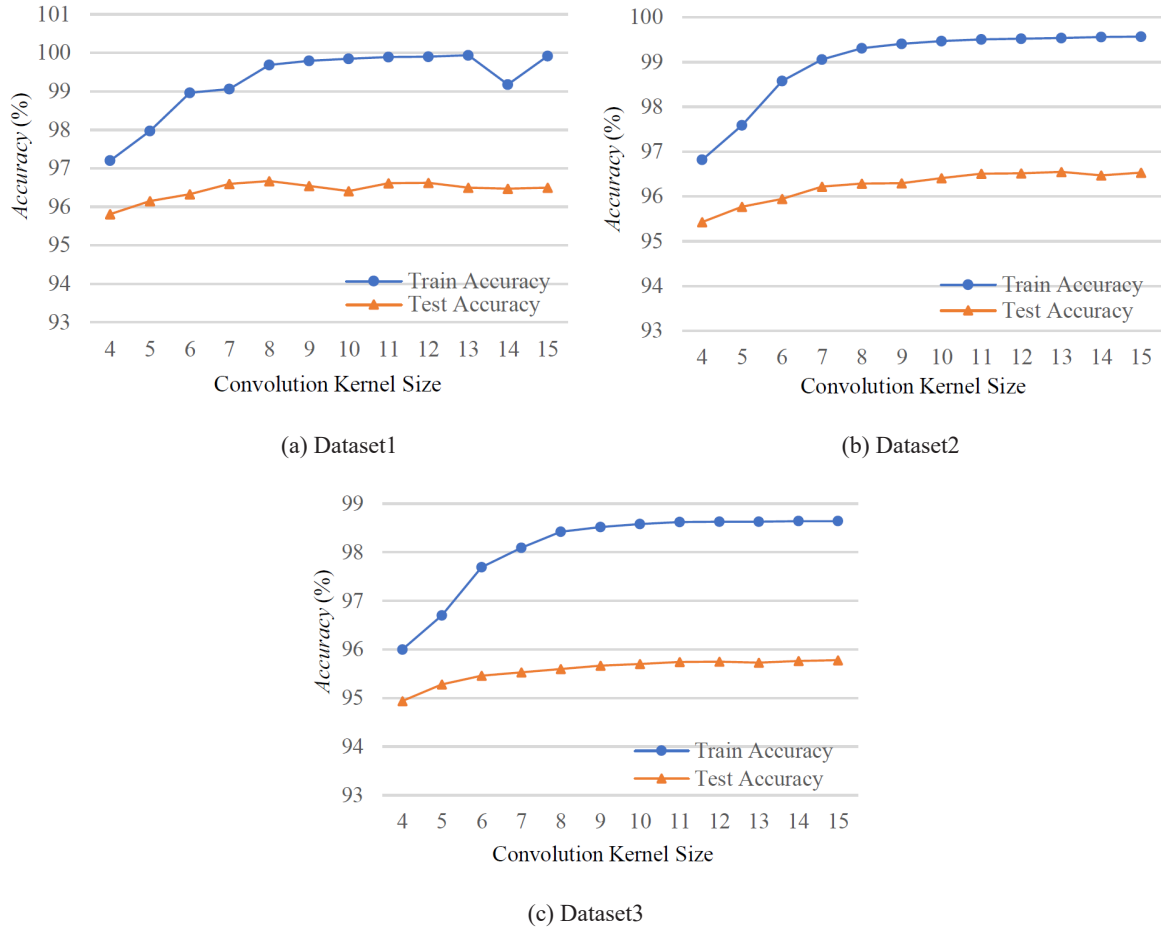


Fig. 9. Impact of different convolutional kernel sizes on the *Accuracy* of the DPMLF model

Depth of Cascaded Dense-connected Convolutional Network. An initial convolutional layer with 32 output channels and a kernel size of $1 \times d$ was used to transform the initial feature dimension from d to 32 to assess the

effect of the depth of the cascaded dense convolutional networks on the model performance. In each subsequent layer, a convolutional layer with 32 output channels and a kernel size of three was employed, followed by a pooling layer with a stride of two to reduce feature redundancy.

In the experiments, the HTML sequence length m was set to 1000 and the word-embedding dimension d was set to 16. The accuracy was evaluated as the number of convolutional blocks increased. As shown in Fig. 10, the accuracy of the model for the three datasets improves significantly as the depth increases from one to three convolutional blocks. However, further increases in the number of convolutional blocks led to diminishing returns, and the performance slightly declined. In Dataset 2, the accuracy of the model showed a relatively significant decline. When the data distribution is complex, or the sample size is limited, an excessive number of convolutional blocks may lead to model overfitting. Therefore, to balance model performance and complexity, a depth of three convolutional blocks was selected for the cascaded dense convolutional network.

4.5 Ablation Experiments

In this section, the influence of diverse input methods on the performance of the model is analyzed via feature ablation experiments. Three input methods were employed in these experiments: utilizing only the URL, using only the HTML content, and simultaneously leveraging both the URL and HTML content.

Tables 2, Table 3, and Table 4 summarize the test results of the DPMLF model on Datasets 1, 2, and 3, respectively, without incremental training. The findings indicate that the detection performance of the model is optimal when it learns from both the URL and HTML features. The URL provides webpage address information, whereas the HTML captures the semantic features of the page. The combination of these two inputs significantly enhanced detection accuracy.

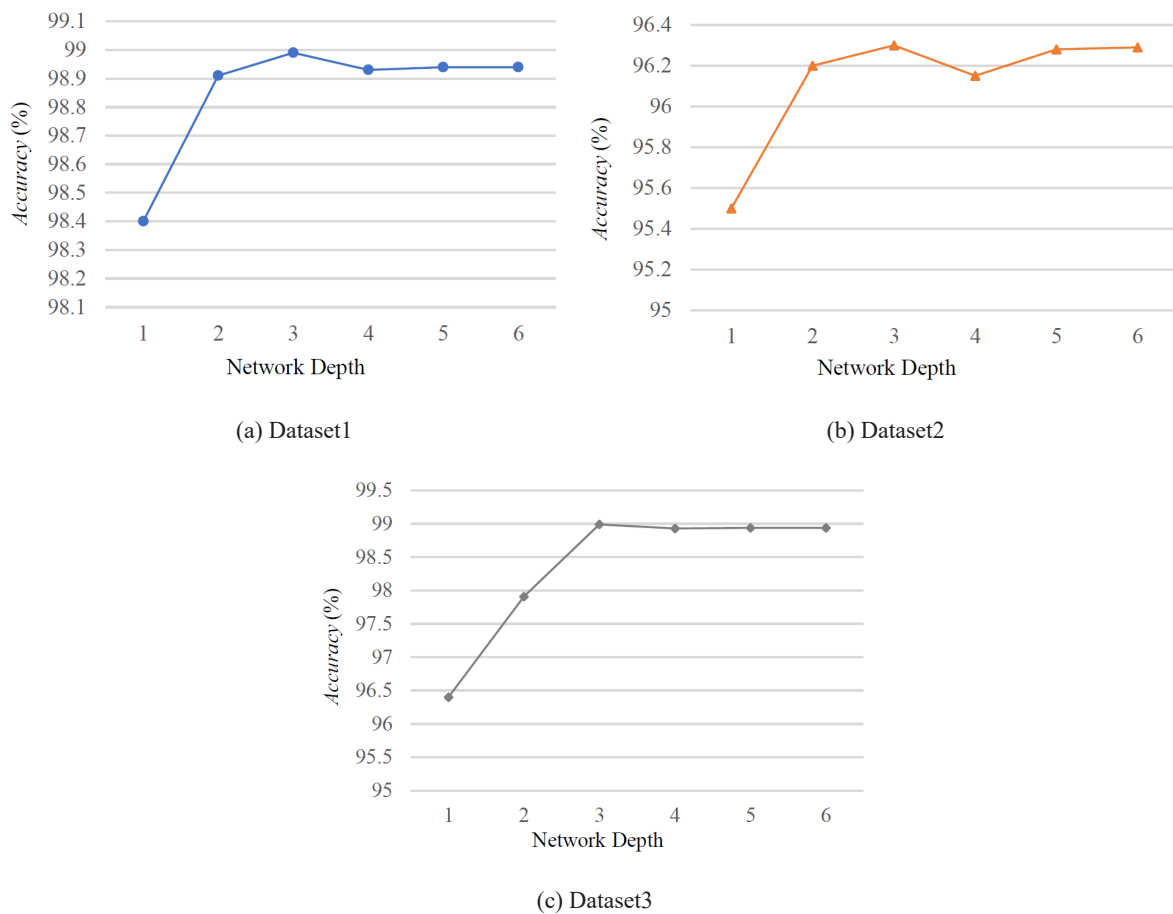


Fig. 10. Impact of network depth on the Accuracy of the DPMLF model

Table 2. Performance evaluation results in Dataset1 (%)

Input	Accuracy	Precision	Recall	F ₁ -score
URL	97.06	96.45	97.20	96.82
HTML	98.98	98.76	99.04	98.90
URL+HTML	99.54	99.34	99.67	99.51

Table 3. Performance evaluation results in Dataset2 (%)

Input	Accuracy	Precision	Recall	F ₁ -score
URL	97.66	97.50	97.80	97.65
HTML	96.37	95.99	96.69	96.34
URL+HTML	99.10	98.74	99.44	99.09

Table 4. Performance evaluation results in Dataset3 (%)

Input	Accuracy	Precision	Recall	F ₁ -score
URL	94.30	94.20	94.40	94.20
HTML	96.87	96.00	97.81	96.90
URL+HTML	98.26	98.59	98.00	98.29

4.6 Model Performance Validation of Incremental Training Method

A progressive incremental training strategy was implemented to enhance the model performance. In this experiment, we first use Dataset1 for pretraining to grasp the data features. Subsequently, we applied the model parameters trained in Dataset1 and Dataset2 for further training and testing to adapt to the new data. Next, we used Dataset3 to train and test the model, and further optimize its performance. Finally, we reapplied the model to Dataset1 for fine-tuning and testing to ensure an improvement in the model performance.

As summarized in Table 5, the incremental learning process resulted in a notable enhancement in the performance of the DPMLF model as the quantity of pretraining data increased.

Table 5. Incremental training the DPMLF (%)

	Dataset	Accuracy	Precision	Recall	F ₁ -score
Dataset1	DPMLF	99.54	99.34	99.67	99.51
	DPMLF _{-Incremental}	99.55	99.34	99.69	99.52
Dataset2	DPMLF	99.10	98.74	99.44	99.09
	DPMLF _{-Incremental}	99.17	99.05	99.29	99.17
Dataset3	DPMLF	98.26	98.59	98.00	98.29
	DPMLF _{-Incremental}	98.28	98.64	98.00	98.31

4.7 Overall Performance of DPMLF

In this section, the DPMLF model is compared with six existing relevant phishing detection models. Among these six models: (1) PhishTrim [44] leverages a skip-gram pretraining model to generate the initial URL embeddings. Subsequently, it deploys a BiLSTM network to capture context-dependent relationships and utilizes different-scale CNNs to extract local features. (2) CNN-fusion [6] adopts a multiscale parallel CNN strategy to extract character combination features from URLs. These features are then classified using a fusion technique. (3) Web2Vec [41] regards URLs, HTML page content, and DOM structures as sequences of words and characters. It employs a hybrid DL network (comprising CNN and BiLSTM) to extract both local and global features. Prior to classification, an attention mechanism is applied to enhance the significant features. (4) WebPhish [7] embeds the original URLs and HTML content, converts word sequences into dense vectors, merges the embedding matrices, and models their semantic dependencies using a convolutional layer. (5) Fuzz-URL [45] is a phishing URL detection model that integrates a transformer network with expert knowledge offered by fuzzy logic, enhancing

its ability to interpret and adapt to the complex patterns of phishing URLs. (6) BiLSTM-URL [35] is a DL model that detects malicious URL based on character- and word-based feature extraction.

Table 6 to Table 8 present the phishing website detection results of these models on the Dataset1, Dataset2, and Dataset3, respectively. The experimental data clearly indicate that the DPMLF model consistently outshines other models in phishing detection. PhishTrim, CNN-Fusion, Fuzz-URL, and BiLSTM-URL, which solely rely on URLs, yield incomplete feature representations and subpar classification results. Web2Vec employs multi-channel CNNs and a BiLSTM to capture more comprehensive information. However, it also incorporates noise from URLs and HTML content, thereby diminishing its accuracy. WebPhish, which combines URL and HTML content embeddings, encounters issues such as redundancy, heightened complexity, and reduced processing efficiency. Moreover, its single-channel CNN proves inadequate for effective feature extraction. In contrast, the DPMLF model attains superior accuracy and efficiency by overcoming these limitations.

Table 6. Phishing detection performance on Dataset1 (%)

Model	Accuracy	Precision	Recall	F ₁ -score
PhishTrim	96.10	94.08	97.41	95.72
CNN-Fusion	97.69	97.03	97.97	97.50
Web2Vec	99.05	98.69	98.26	99.08
WebPhish	98.81	98.38	99.03	98.70
Fuzz-URL	98.05	97.52	96.65	97.56
BiLSTM-URL	97.92	98.22	97.92	97.86
DPMLF _{-Incremental}	99.55	99.34	99.69	99.52

Table 7. Phishing detection performance on Dataset2 (%)

Model	Accuracy	Precision	Recall	F ₁ -score
PhishTrim	96.14	95.87	96.09	95.98
CNN-Fusion	97.82	98.43	97.22	97.82
Web2Vec	97.98	97.43	98.21	97.82
WebPhish	98.10	98.20	98.10	98.10
Fuzz-URL	97.67	97.36	96.23	96.78
BiLSTM-URL	97.33	97.45	97.34	97.23
DPMLF _{-Incremental}	99.17	99.05	99.29	99.17

Table 8. Phishing detection performance on Dataset3 (%)

Model	Accuracy	Precision	Recall	F ₁ -score
PhishTrim	94.27	95.50	92.91	94.21
CNN-Fusion	94.72	94.34	95.14	94.72
Web2Vec	96.80	96.72	96.88	96.76
WebPhish	97.56	97.53	97.59	97.58
Fuzz-URL	96.25	96.02	95.88	96.13
BiLSTM-URL	95.54	95.12	94.88	94.92
DPMLF _{-Incremental}	98.28	98.64	98.00	98.31

5 Conclusion

This study proposed a novel deep phishing detection model named DPMLF that seamlessly integrated the semantics of URL characters and HTML content. The model accurately differentiated phishing pages from legitimate pages by leveraging multilevel and fine-grained semantic features, thereby eliminating the need for expert knowledge or third-party services. Specifically, the model deployed parallel CNNs to capture diverse semantic information embedded within URL character sequences comprehensively. Concurrently, it utilized a cascaded densely connected CNN to perform multiscale analysis of HTML long texts, effectively extracting long-range textual information. Furthermore, incremental learning was incorporated into the training process. This approach

enabled continuous updating of the knowledge base using newly acquired datasets, ensuring that the model remained adaptive and effective in the face of evolving phishing threats. Future studies will focus on exploring the integration of Web image information with semantic features. This promising avenue is expected to further enhance the performance and stability of the model, paving the way for more robust phishing detection solutions in the digital landscape.

Acknowledgement

This study was supported by the University Natural Science Research Project of Anhui Province (China) (No. 2024AH050617, 2024AH050612) and the Natural Science Foundation of Anhui Xinhua University (China) (No. 2023ZR001).

References

- [1] G. Varshney, R. Kumawat, V. Varadharajan, U. Tupakula, C. Gupta, Anti-phishing: A comprehensive perspective, *Expert Systems with Applications* 238(2024) 122199. <https://doi.org/10.1016/j.eswa.2023.122199>
- [2] APWG, Phishing activity trends Reports: 1Q~4Q 2024. <<https://apwg.org/trendsreports/>>, 2024. (accessed 03.19.2025).
- [3] E.-Z. Zhu, K. Cheng, Z.-Z. Zhang, H. Wang, PDHF: Effective Phishing Detection Model Combining Optimal Artificial and Automatic Deep Features, *Computers & Security* 136(2024) 103561. <https://doi.org/10.1016/j.cose.2023.103561>
- [4] R.-F. Liu, Y. Lin, X.-L. Yang, S.H. Ng, Inferring phishing intention via webpage appearance and dynamics: a deep vision based approach, in: *Proc. 31st USENIX Security Symposium*, 2022. <https://www.usenix.org/conference/usenixsecurity22/presentation/liu-ruofan>
- [5] Y.-J. Liang, Q.-S. Wang, K. Xiong, X. Zheng, Z. Yu, D. Zeng, Robust detection of malicious URLs with self-paced wide & deep learning, *IEEE Transactions on Dependable and Secure Computing* 19(2)(2022) 717-730. <https://doi.org/10.1109/TDSC.2021.3121388>
- [6] M. Hussain, C. Cheng, R. Xu, M. Afzal, CNN-Fusion: An effective and lightweight phishing detection method based on multi-variant ConvNet, *Information Sciences* 631(2023) 328-345. <https://doi.org/10.1016/j.ins.2023.02.039>
- [7] C. Opara, Y.-K. Chen, B. Wei, Look before you leap: Detecting phishing web pages by exploiting raw URL and HTML characteristics, *Expert Systems with Applications* 236(2024) 121183. <https://doi.org/10.1016/j.eswa.2023.121183>
- [8] D.-J. Liu, G.-G. Geng, X.-C. Zhang, Multi-scale Semantic Deep Fusion Models for Phishing Website Detection, *Expert Systems with Applications* 209(2022) 118305. <https://doi.org/10.1016/j.eswa.2022.118305>
- [9] Google, Google Safe Browsing. <<http://code.google.com/apis/safebrowsing/>> (accessed 01.18.2025).
- [10] A. Oest, Y. Safaei, P.-H. Zhang, B. Wardman, K. Tyers, Y. Shoshitaishvili, A. Doupé, G.-J. Ahn, PhishTime: Continuous Longitudinal Measurement of the Effectiveness of Anti-phishing Blacklists, in: *Proc. 19th USENIX Security Symposium*, 2020. <https://www.usenix.org/conference/usenixsecurity20/presentation/oest-phishime>
- [11] I. Skula, M. Kvet, Domain Blacklist Efficacy for Phishing Web-page Detection Over an Extended Time Period, in: *Proc. 33rd Conference of Open Innovations Association*, 2023. <https://doi.org/10.23919/FRUCT58615.2023.10142999>
- [12] R. Zaimi, K. S. Eljil, M. Hafidi, M. Lamia, F. Nait-Abdesselam, An enhanced mechanism for malicious URL detection using deep learning and DistilBERT-based feature extraction, *Journal of Supercomputing* 81(2)(2025) 438. <https://doi.org/10.1007/s11227-024-06908-x>
- [13] C. Silva, E. Feitosa, V. Garcia, Heuristic-based strategy for Phishing prediction: A survey of URL-based approach, *Computers & Security* 88(2020) 101613. <https://doi.org/10.1016/j.cose.2019.101613>
- [14] S. Shukla, M. Misra, G. Varshney, HTTP header based phishing attack detection using machine learning, *Transactions on Emerging Telecommunications Technologies (Wiley)* 35(1)(2024) 1-22. <https://doi.org/10.1002/ett.4872>
- [15] R.-S. Rao, A. Pais, P. Anand, A heuristic technique to detect phishing websites using TWSVM classifier, *Neural Computing and Applications* 33(11)(2021) 5733-5752. <https://doi.org/10.1007/s00521-020-05354-z>
- [16] J. Chen, S. Mishler, B. Hu, Automation Error Type and Methods of Communicating Automation Reliability Affect Trust

- and Performance: An Empirical Study in the Cyber Domain, *IEEE Transactions on Human-Machine Systems* 51(5) (2021) 463-473.
<https://doi.org/10.1109/THMS.2021.3051137>
- [17] S.-J. Li, G.-P. Gou, C. Liu, C. Hou, Z. Lin, G. Xiong, TTAGN: temporal transaction aggregation graph network for ethereum phishing scams detection, in: *Proc. 31st ACM World Wide Web Conference*, 2022.
<https://doi.org/10.1145/3485447.3512226>
- [18] J.-L. Liu, J.-Z. Chen, J.-J. Wu, Z. Wu, J. Fang, Z. Zheng, Fishing for Fraudsters: Uncovering Ethereum Phishing Gangs with Blockchain Data, *IEEE Transactions on Information Forensics and Security* 19(2024) 3038-3050.
<https://doi.org/10.1109/TIFS.2024.3359000>
- [19] J.-T. Luo, J.-W. Qin, R.-J. Wang, L. Li, A Phishing Account Detection Model via Network Embedding for Ethereum, *IEEE Transactions on Circuits and Systems II: Express Briefs* 71(2)(2024) 622-626.
<https://doi.org/10.1109/TCSII.2023.3267822>
- [20] B. Zhang, Y.-S. Gao, B.-Y. Kuang, C. Yu, A. Fu, W. Susilo, A Survey on Advanced Persistent Threat Detection: A Unified Framework, Challenges, and Countermeasures. *ACM Computing Surveys* 57(3)(2025) 62.
<https://doi.org/10.1145/3700749>
- [21] M. Khonji, Y. Iraqi, A. Jones, Phishing detection: a literature survey, *IEEE Communications Surveys and Tutorials* 15(4)(2013) 2091-2121.
<https://doi.org/10.1109/SURV.2013.032213.00009>
- [22] E.-Z. Zhu, Y.-X. Zhang, P. Wen, F. Liu, Fast and Stable Clustering Analysis based on Grid-mapping K-means Algorithm and New Clustering Validity Index, *Neurocomputing* 363(2019) 149-170.
<https://doi.org/10.1016/j.neucom.2019.07.048>
- [23] S. Abdelnabi, K. Kromholz, M. Fritz, VisualPhishNet: Zero-Day phishing website detection by visual similarity, in: *Proc. 27th ACM SIGSAC Conference on Computer and Communications Security*, 2020.
<https://doi.org/10.1145/3372297.3417233>
- [24] M.-L. Wang, L.-P. Song, L.-Y. Li, Y.-H. Zhu, J. Li, Phishing webpage detection based on global and local visual similarity, *Expert Systems with Applications* 252(2024) 124120.
<https://doi.org/10.1016/j.eswa.2024.124120>
- [25] C.-C. Tan, K.-L. Chiew, K.-S. Yong, Y. Sebastian, J.C.M. Than, W.K. Tiong, Hybrid phishing detection using joint visual and textual identity, *Expert Systems with Applications* 220(2023)119723.
<https://doi.org/10.1016/j.eswa.2023.119723>
- [26] W.-P. Wen, Y.-F. Zhu, Z.-H. Lv, C.-J. Liu, Brand-Specific Phishing Expansion and Detection Solutions, *Netinfo Security* 23(12)(2023) 1-9.
<http://dx.doi.org/10.3969/j.issn.1671-1122.2023.12.001>
- [27] R. Liu, Y. Lin, X. Teoh, G. Liu, Z. Huang, J.S. Dong, Less defined knowledge and more true alarms: reference-based phishing detection without a pre-defined reference list, in: *Proc. 33rd USENIX Security Symposium*, 2024.
<https://www.usenix.org/conference/usenixsecurity24/presentation/liu-ruofan>
- [28] M. Bahaghighat, M. Ghasemi, F. Ozen, A high-accuracy phishing website detection method based on machine learning, *Journal of Information Security and Applications* 77(2023) 103533.
<https://doi.org/10.1016/j.jisa.2023.103553>
- [29] C. Silva, B. Fernandes, E. Feitosa, V. Garcia, Piracema.io: A rules-based tree model for phishing prediction, *Expert Systems with Applications* 191(2022) 116239.
<https://doi.org/10.1016/j.eswa.2021.116239>
- [30] E.-Z. Zhu, Z.-L. Chen, J. Cui, H. Zhong, MOE/RF: A Novel Phishing Detection Model Based on Revised Multiobjective Evolution Optimization Algorithm and Random Forest, *IEEE Transactions on Network and Service Management* 19(4)(2022) 4461-4478.
<https://doi.org/10.1109/TNSM.2022.3162885>
- [31] A. Kulkarni, V. Balachandran, T. Das, Phishing Webpage Detection: Unveiling the Threat Landscape and Investigating Detection Techniques, *IEEE Communications Surveys and Tutorials* 27(2)(2025) 974-1007.
<https://doi.org/10.1109/COMST.2024.3441752>
- [32] D. Panneerselvam, S.C. Sethuraman, A.J. Emerson, T.K. Kanakam, A Concise Survey on Modern Web-Based Phishing Techniques and Advanced Mitigation Strategies, *Transactions on Emerging Telecommunications Technologies* 36(4) (2025) e70119.
<https://doi.org/10.1002/ett.70119>
- [33] T.-K. Wen, Y.-X. Xiao, A.-Q. Wang, H. Wang, A novel hybrid feature fusion model for detecting phishing scam on Ethereum using deep neural network, *Expert Systems with Applications* 211(2023) 118463.
<https://doi.org/10.1016/j.eswa.2022.118463>
- [34] D.-J. He, X. Lv, S. Zhu, S. Chan, K.-K.R. Choo, A Method for Detecting Phishing Websites Based on Tiny-Bert Stacking, *IEEE Internet of Things Journal* 11(2)(2024) 2236-2243.
<https://doi.org/10.1109/JIOT.2023.3292171>
- [35] O.S. Akcam, A. Tekerek, M. Tekerek, Development of BiLSTM deep learning model to detect URL-based phishing attacks, *Computers & Electrical Engineering* 123(PartC)(2025) 110212.
<https://doi.org/10.1016/j.compeleceng.2025.110212>

- [36] S.K. Birthriya, P. Ahlawat, A.K. Jain, Detection and prevention of spear phishing attacks: A comprehensive survey, *Computers & Security* 151(2025) 104317.
<https://doi.org/10.1016/j.cose.2025.104317>
- [37] B.-F. Li, A. Drozd, Y.-H. Guo, T. Liu, S. Matsuoka, X. Du, Scaling Word2Vec on Big Corpus, *Data Science and Engineering* 4(2)(2019) 157-175.
<https://doi.org/10.1007/s41019-019-0096-6>
- [38] A. Pimpalkar, R.J. Retna, MBiLSTMGloVe: Embedding GloVe knowledge into the corpus using multi-layer BiLSTM deep learning model for social media sentiment analysis, *Expert Systems with Applications* 203(2022) 117581.
<https://doi.org/10.1016/j.eswa.2022.117581>
- [39] S.-Y. Wang, M.-L. Huang, Z.-D. Deng, Densely Connected CNN with Multi-scale Feature Attention for Text Classification, in *Proc. 27th International Joint Conference on Artificial Intelligence*, 2018.
<https://doi.org/10.24963/ijcai.2018/621>
- [40] A. Prasad, S. Chandra, PhiUSiL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning, *Computers & Security* 136(2024) 103545.
<https://doi.org/10.1016/j.cose.2023.103545>
- [41] J. Feng, L.-Y. Zou, O. Ye, Z. Han, Web2Vec: Phishing Webpage Detection Method Based on Multidimensional Features Driven by Deep Learning, *IEEE Access* 8(2020) 221214-221224.
<https://doi.org/10.1109/ACCESS.2020.3043188>
- [42] M. Al-Maamari, M. Istaiti, S. Zerhoudi, M. Dinzinger, M. Granitzer, J. Mitrovic, A Comprehensive Dataset for Webpage Classification. <<https://doi.org/10.5281/zenodo.10795437>>, 2024 (accessed 18.11.2024).
- [43] A.S. Bozkir, F.C. Dalgic, M. Aydos, GramBeddings: A New Neural Network for URL Based Identification of Phishing Web Pages Through N-gram Embeddings, *Computers & Security* 124(2023) 102964.
<https://doi.org/10.1016/j.cose.2022.102964>
- [44] L. Zhang, P. Zhang, PhishTrim: Fast and adaptive phishing detection based on deep representation learning, in: *Proc. 13th IEEE International Conference on Web Services*, 2020.
<https://doi.org/10.1109/ICWS49710.2020.00030>
- [45] S.J. Buu, S.-B. Cho, A Transformer network calibrated with fuzzy logic for phishing URL detection, *Fuzzy Sets and Systems* 517(2025) 109474.
<https://doi.org/10.1016/j.fss.2025.109474>