scientific reports

Check for updates

OPEN

An integrated CSPPC and BiLSTM framework for malicious URL detection

Jinyang Zhou, Kun Zhang✉, Anas Bilal✉, Yu Zhou✉, Yukang Fan, Wenting Pan, Xin Xie & Qi Peng

With the rapid development of the internet, phishing attacks have become more diverse, making phishing website detection a key focus in cybersecurity. While machine learning and deep learning have led to various phishing URL detection methods, many remain incomplete, limiting accuracy. This paper proposes CSPPC-BiLSTM, a malicious URL detection model based on BiLSTM (Bidirectional Long Short-Term Memory, BiLSTM). The model processes URL character sequences through an embedding layer and captures contextual information via BiLSTM. By integrating CBAM (Convolutional Block Attention Module, CBAM), it applies channel and spatial attention to highlight key features and transforms URL sequence features into a spatial matrix. The SPP (Spatial Pyramid Pooling, SPP) module enables multi-scale pooling. Finally, a fully connected layer fuses features, and dropout regularization enhances robustness. Compared to CharBiLSTM, CSPPC-BiLSTM significantly improves detection accuracy. Evaluated on two datasets, Grambedding (balanced) and Mendeley AK Singh 2020 phish (imbalanced)—and compared with six baselines, it demonstrates strong generalization and accuracy. Ablation experiments confirm the critical role of CBAM and SPP in boosting performance.

Keywords Phishing, Malicious URL detection, Deep learning, BiLSTM, CBAM, SPP

Malicious URLs are one of the primary mechanisms for committing cybercrimes in the form of deceptive web links. These URLs are created to disseminate spam, phishing schemes, malware, ransomware, spyware, and more. Users may inadvertently download malware by clicking infected links, adversely affecting their systems, or may be tricked into providing confidential information to fraudulent websites, leading to significant losses[1].

According to recent reports, phishing attacks remain a major threat. The 2023 Internet Crime Report by the IC3 (Internet Crime Complaint Center, IC3) of the U.S. Federal Bureau of Investigation revealed 298,878 reported phishing-related complaints in 2023, with BEC (Bsiness Email Compromise, BEC) and phishing collectively causing losses of up to $2.9 trillion. The 2024 Phishing Report by Zscaler ThreatLabz also highlights a nearly 60% year-on-year increase in phishing attacks globally in 2023 compared to 2022. Among these, recruitment scams, vishing (voice phishing), and malicious URL-based attacks have surged significantly. As the cyber landscape evolves, attackers continue to refine and innovate their techniques. Thus, there is an urgent need for more efficient and accurate adaptive technologies to detect malicious URLs effectively.

In terms of classification methods, Vanhoenshoven F. and Nápoles G[2]. applied various machine learning algorithms such as Support Vector Machines, Logistic Regression, and Random Forests to detect malicious phishing URLs and statistically compared their performances. However, machine learning methods often rely heavily on feature engineering and fail to capture the semantic relationships between characters within URLs. Given the continuous evolution of attack tools and the increasing sophistication of attackers, the effectiveness of traditional machine learning approaches in identifying malicious phishing URLs has become increasingly limited.

In the field of malicious URL detection, numerous studies leveraging machine learning and deep learning have achieved significant progress. For instance, Wei Wei, Ke Q. and colleagues[3] proposed an anti-phishing system based on CNN (Convolutional Neural Networks, CNN), demonstrating the strong performance of CNNs in detecting malicious phishing URLs. Another study[4] I ntroduced URLdeepDetect, a semantic vector-based detection model that combines LSTM (Long Short-Term Memory, LSTM) networks with k-means clustering to perform supervised and unsupervised classification of malicious URLs. Wang H., Yu L., and colleagues[5] proposed a detection approach based on the CBIR algorithm, extracting "texture fingerprint" features representing binary file content similarity and integrating them with URL word vector, host information, and

School of Information Science and Technology, Hainan Normal University, Haikou 571158, Hainan, China. ✉email: kunzhang@hainnu.edu.cn; 910288@hainnu.edu.cn; zhouyu@hainnu.edu.cn

URL structural features to improve detection accuracy. Other notable works include URLNet[6], which pioneered the combination of dual-path CNNs with semantic feature understanding, utilizing a Text-CNN approach for malicious URL detection. A deep learning-based detection system called CyberLen[7] was developed to address sample complexity and diversity issues through heterogeneous feature fusion, improving model robustness. The DA-BiGRU model[8] combined Bidirectional GRU and attention mechanisms for malicious URL detection, employing Word2Vec to train URL word vectors and introducing regularization in the input layer to reduce overfitting. Nanda M. and colleagues[9] proposed a hybrid model combining character-level and word-level embeddings, attention mechanisms, BiLSTM, and CNN. Another high-performance model, PMANet[10], utilized PLM (Pre-trained Language Models, PLM) and advanced Transformer networks for malicious URL detection. Similarly, Mahdaouy A. E. and colleagues[11] introduced DomURLs_BERT, a pre-trained model on a large-scale multilingual corpus containing URLs, domain names, and domain generation algorithm datasets, specifically designed for malicious domain and URL classification.

Ehsan Nowroozi[12] adopted Random Forest, Gradient Boosting, XGBoost, and AdaBoost classifiers, training and testing across diverse datasets to detect, predict, and classify malicious ad URLs. Erzhou Zhu[13] developed a lightweight phishing detection model, CCBLA, based on CNN, BiLSTM, and attention mechanisms, demonstrating its effectiveness and efficiency through experiments. Finally, Yun-Da Tsai[14] highlighted the issue of data bias in machine learning models for malicious URL detection, which can significantly impact model performance. The Jeeva S C team[15] performed correlation rule mining on URL features to detect them. Lee[1] O V et al. used optimization and machine learning to detect malicious URLs and evaluate their effectiveness[16]. AlEroud and Karabatis[17] propose a method to generate URL-based phishing examples using GANs (Generative Adversarial Neural Networks, GANs) to bypass existing URL-based phishing detection methods. Karajgar[18] solves the problem of comparing the effectiveness of different machine learning models in identifying malicious URLs by evaluating multiple machine learning models on a dataset of more than 700,000 URLs. Ozcan[19] proposed a hybrid deep learning model to detect phishing URLs, which solved the problem of phishing detection accuracy. Remya[20] proposed an effective method to detect phishing URLs using residual pipes, which solved the problem that dynamic URLs were difficult to identify by traditional blacklists and improved the accuracy of phishing threat detection. Islam[21] solves the problem of using machine learning to detect malicious URLs to keep users online. Mia[22] analyzed two public phishing URL datasets and used Interpretable Artificial Intelligence to explore the trustworthiness of phishing URL detection features across different datasets, which solved the problem of whether phishing URL detection features are common across datasets. Kaushik[23] proposed a deep learning-based phishing attack detection method to solve the problem of detecting phishing attacks in the field of network security. Su[24] proposed a BERT-based method to identify malicious URLs, demonstrating the effectiveness of the method in detecting malicious URLs through experiments on three different public datasets. Geyik[25] solved the problem of using URL detection to detect phishing websites by testing the public dataset CatchPhish 3 using different classification techniques on WEKA. Taofeek[26] proposed a new machine learning-based phishing detection method, which solves the problem that traditional detection methods are difficult to keep up with the evolution of attackers' techniques and fail to protect users.

Using interpretable machine learning techniques, the study proposed an adversarial training strategy to mitigate bias in deep learning models, enhancing the generalization capabilities of CNN and RNN-based detection models, as validated by experimental results.

To further enhance the accuracy and efficiency of phishing malicious URL detection, this paper proposes a novel detection technique based on the CSPPC-BiLSTM model.

The main contributions of this study are summarized as follows:

(1) The CSPPC-BiLSTM model integrates BiLSTM, CBAM and SPP technology to convert URLs, which are essentially character sequence data, into multi-dimensional spatial representation for feature extraction. On the one hand, this method makes fusion innovation based on traditional feature extraction methods. On the other hand, it solves the problem of low efficiency of LSTM in processing long sequences and significantly improves the detection accuracy of malicious URLs.
(2) Even in scenarios with highly imbalanced positive and negative sample distributions, the CSPPC-BiLSTM model maintains high accuracy in detecting malicious URLs, demonstrating strong robustness.

Related work

Bidirectional long short-term memory

In 1997, Mike Schuster and Kuldip K. Paliwal[27] first introduced the concept of Bidirectional Recurrent Neural Networks (BRNN). In the same year, Hochreiter and Schmidhuber[28] proposed the Long Short-Term Memory (LSTM) model to address the vanishing gradient problem in traditional Recurrent Neural Networks (RNNs), which makes it difficult to capture long-term dependencies.

Related work BiLSTM is a specialized type of RNN that combines the principles of BRNN and LSTM. BiLSTM can process sequential data while preserving long-term dependencies. Unlike traditional RNNs, BiLSTM considers both past and future information simultaneously, enabling it to capture contextual relationships in sequential data more effectively. In essence, BiLSTM consists of two independent LSTM networks: one processes the sequence in the forward direction, and the other processes it in reverse. The structure of BiLSTM is illustrated in Fig. 1.

Forget gate

Figure 2 highlights the structure of the forget gate in LSTM, shown within the red box. The forget gate determines which information to discard from the cell state. It takes the previous output h_{t-1} and the current input x_t, applies a non-linearσ (Sigmoid) activation, and outputs a vector f_t with values ranging between 0 and 1. This vector is

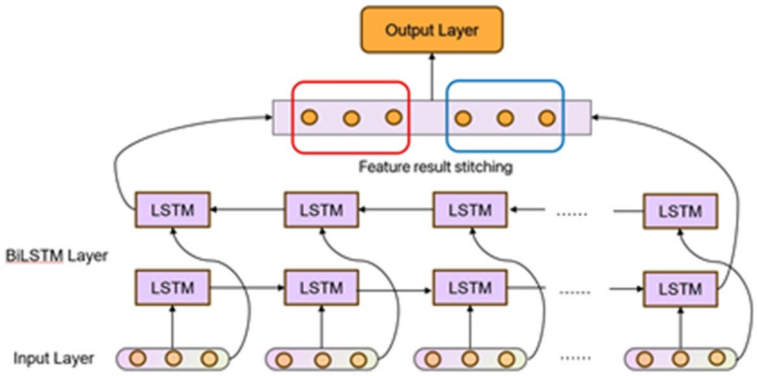

Fig. 1. Structure of BiLSTM.

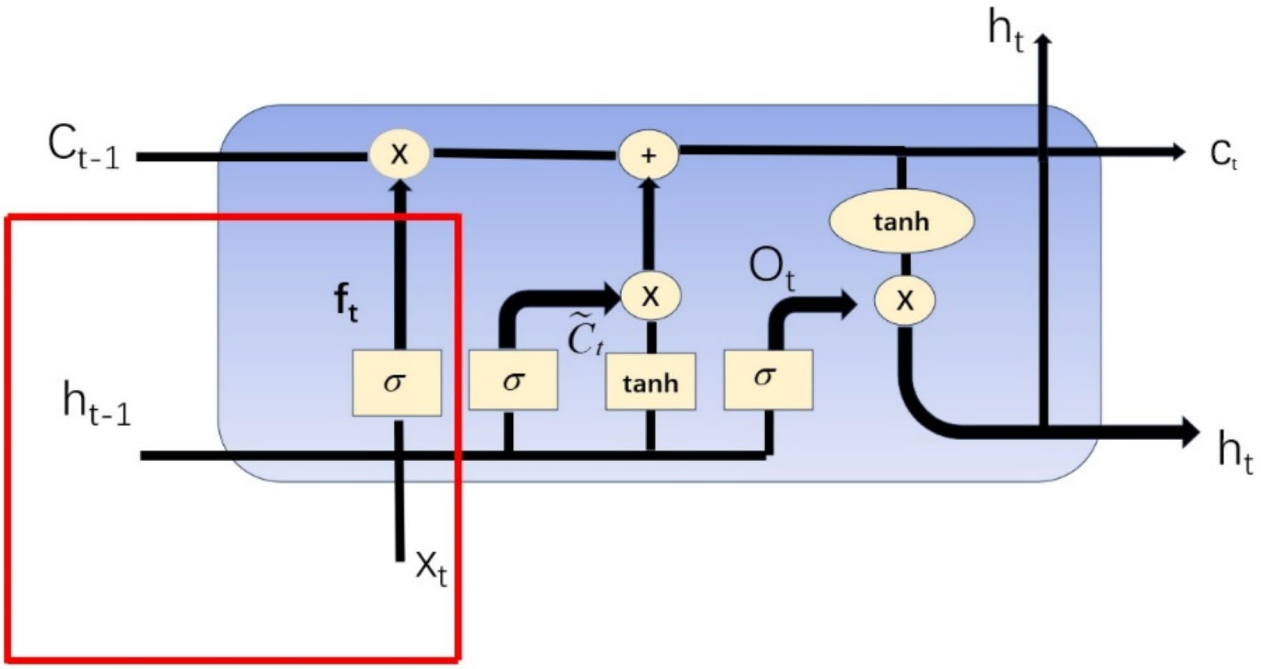

Fig. 2. Forget gate structure.

then multiplied with the cell state to selectively forget certain information. The calculation for the forget gate is shown in Eq. (1).

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{1}$$

Input gate

Figure 3 highlights the structure of the input gate in LSTM, shown within the red box. The input gate determines how much new information will be stored in the memory cell at the current time step. It generates values between 0 and 1 based on the current input and the previous hidden state, controlling which information should be added to the memory.

The input gate consists of two parts: the first determines what content to update, as shown in Eq. (2), and the second generates a candidate memory vector, as shown in Eq. (3), which represents new potential information:

I_t: The output of the input gate is a vector. The value of each element is between 0 and 1.

W_i: The weight matrix of the input gate, which maps the hidden state of the previous moment and the current input to the output space of the input gate.

b_i: Enter the offset term of the gate to adjust the output value of the input gate.

$\widetilde{C}_t$: The candidate memory content is a vector. Represents the underlying information of the current moment, i.e., the memory content that is ready to be updated.

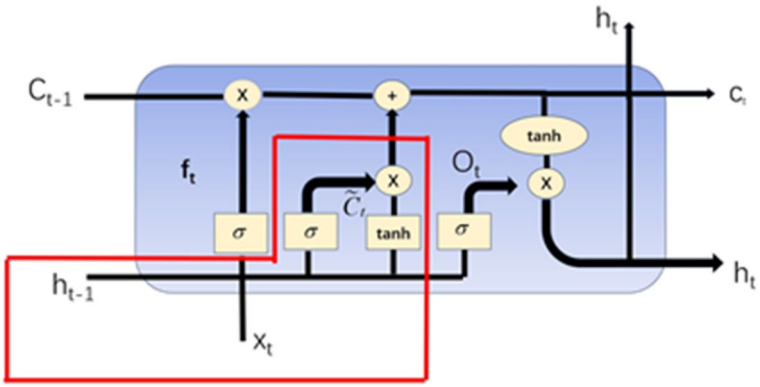

Fig. 3. Input gate structure.

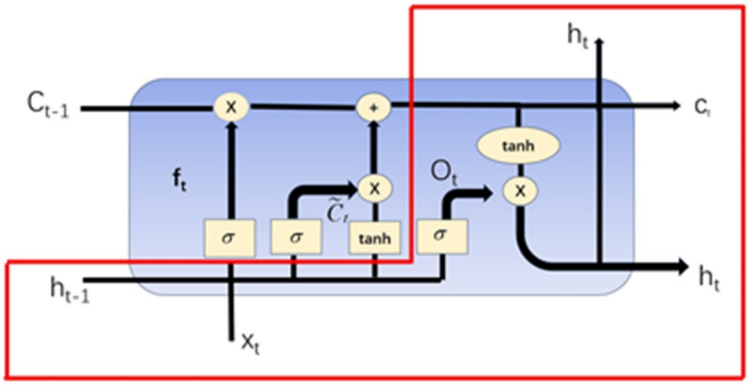

Fig. 4. Output gate structure.

W_C: A weighted matrix of candidate memory content, which is used to map the hidden state of the previous moment and the current input to the space of the candidate memory.

b_C: A bias term for the content of a candidate memory, which is used to adjust the value of the candidate memory.

$$I_t = \sigma(w_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$

$$\widetilde{C}_t = \tanh(w_C \cdot [h_{t-1}, x_t] + b_C) \tag{3}$$

Output gate

The output gate of BiLSTM is a key mechanism that controls the output of the LSTM unit, as shown within the red box in Fig. 4. In LSTM, the output gate determines which information will be output at the current time step. The calculation of the output gate is shown in Eqs. (4) and (5).

In Eq. (4), O_t represents the activation value of the output gate; W_o and b_o are the weight matrix and bias term of the output gate, respectively; h_{t-1} is the hidden state from the previous time step; x_t is the current input. The output gate controls the value between 0 and 1 using the sigmoid activation function.

In Eq. (5), the output gate activation value O_t is combined with the current memory state c_t to generate the final hidden state h_t.

$$O_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{4}$$

$$h_t = o_t \cdot \tanh(c_t) \tag{5}$$

Convolutional block attention module

The attention mechanism significantly enhances the performance of deep neural networks by selectively focusing on important features. CBAM is a lightweight attention mechanism that combines Channel Attention and Spatial Attention to capture more comprehensive contextual feature information. It refines the features adaptively by multiplying the attention map with the input feature map[29]. The structure of CBAM is illustrated in Fig. 5.

The input to CBAM is a convolutional feature map $F \in R^{C \times H \times W}$, where C, H and W represent the number of channels, height, and width, respectively. After passing through the Channel Attention module and the Spatial

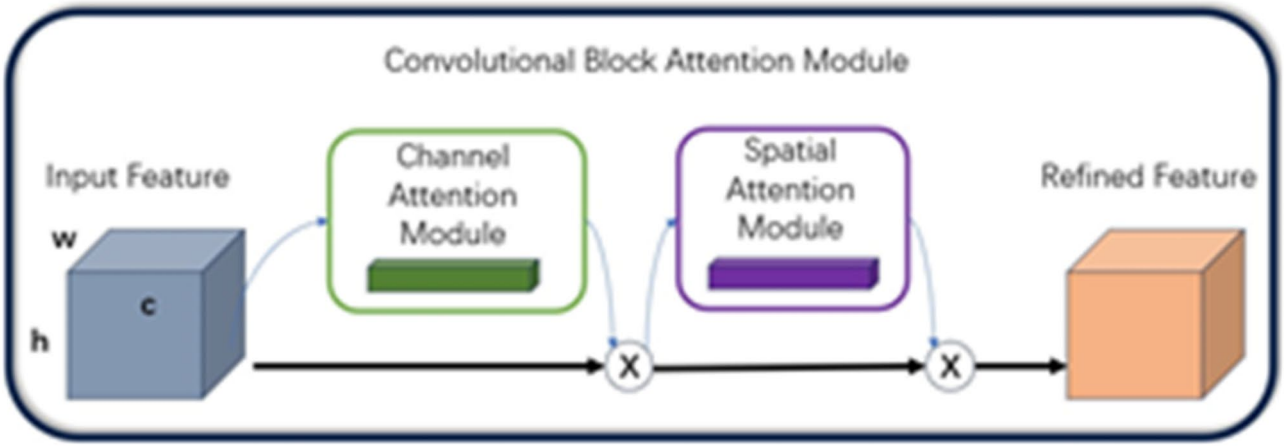

Fig. 5. CBAM structure.

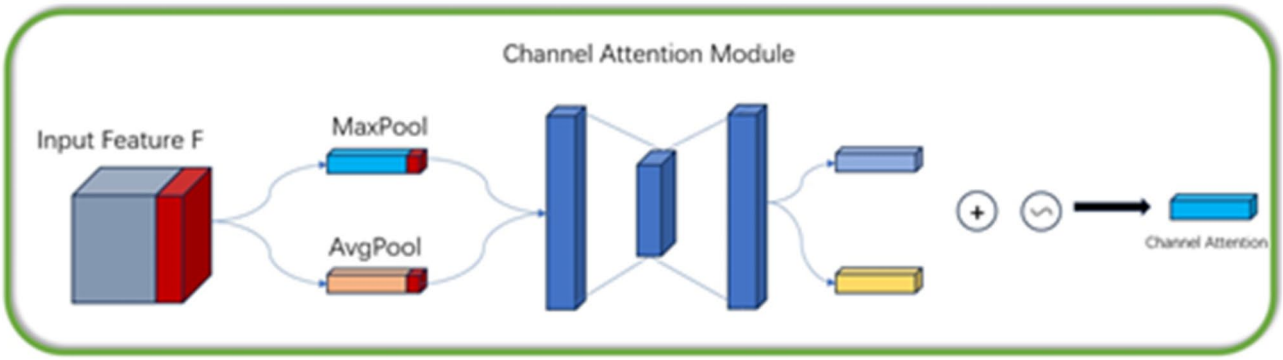

Fig. 6. Channel attention module structure.

Attention module, the weighted feature map F', is obtained, as shown in Eq. (6). Here, M_c (Fc) denotes the channel attention weight, M_s(Fc) represents the spatial attention weight, and ⊙denotes element-wise multiplication.

$$F' = M_s(F_c) \odot M_c(F) \odot F \tag{6}$$

Channel attention module

The channel attention mechanism splits a W×H×C feature into W×H smaller 1 × 1×C feature maps. Each small feature map undergoes both MaxPool and AvgPool operations, followed by a shared-weight operation. The resulting outputs are then summed to produce a 1 × 1×C channel attention map, which is multiplied with the input feature map (W×H×C) to obtain the transformed feature map, also of size W×H×C, as shown in Fig. 6.

The channel attention module assigns weights to each channel to identify the important channels in the feature map. The specific computation steps are as follows:

1. Perform global average pooling and global max pooling on the input feature map F, resulting in two global descriptor vectors $F_{\text{avg}} \in R^C$ and $F_{\max} \in R^C$, respectively:

$$F^{\text{c}}_{\text{avg}} = \frac{1}{H \cdot W} \sum_{\text{i}=1}^{H} \sum_{\text{j}=1}^{W} F(i, j, c) \tag{7}$$

$$F^{\text{c}}_{\max} = \max_{i,j} F(i, j, c) \tag{8}$$

2. As these two vectors through a shared fully connected network (consisting of two fully connected layers):

$$M_c(F) = \sigma(W_1 \cdot \text{Re}\, LU(W_0 \cdot F_{avg}) + W_1 \cdot \text{Re}\, LU(W_0 \cdot F_{\max})) \tag{9}$$

W_0 and W_1 are the weight matrices, and σ is the Softmax activation function.

3. The computed channel attention weights $M_c(F)$ are applied to each channel of the original feature map:

$$F_c = M_c(F) \cdot F \tag{10}$$

Spatial attention mechanisms

The spatial attention mechanism splits the input W×H×C feature map into Cseparate W×H×1 feature maps. Each of these is processed through MaxPool and AvgPool. The resulting pooled feature maps are then summed to obtain a W×H×2 feature map. This feature map undergoes a convolution operation to produce a W×H×1 feature map, which is then multiplied with the original input feature map to obtain the transformed feature map W×H×C, as shown in Fig. 7.

The spatial attention module further identifies the importance of each spatial location in the feature map. The specific computation steps are as follows:

1. Perform global average pooling and global max pooling on the feature map Fc (after the channel attention module) along the channel dimension, resulting in two single-channel feature maps, and$F^s_{\max}$:

$$F^s_{avg}(i,j) = \frac{1}{C}\sum_{c=1}^{C} F_c(i,j,c) \tag{11}$$

$$F^s_{\max}(i,j) = \max_c F_c(i,j,c) \tag{12}$$

2. Concatenate the two feature maps along the channel dimension and pass them through a 7 × 7 convolutional layer:
3. The computed spatial attention weights Ms(Fc) are applied to each spatial location of Fc:

$$M_s(F_c) = \sigma(Conv_{7\times7}([F^s_{avg}, F^s_{\max}])) \tag{13}$$

$$F' = M_s(F_c) \odot F_c \tag{14}$$

Spatial pyramid pooling

Spatial Pyramid Pooling (SPP) is a feature pooling method that performs multi-scale partitioned pooling on a feature map to generate fixed-length feature vectors. It then extends pixel-level features into specifically designed global pyramid-pooling features. By combining local and global cues[30], the final prediction becomes more reliable. SPP is independent of the input image size, enabling it to handle inputs of arbitrary dimensions while preserving spatial information at different scales. The structure of SPP is illustrated in Fig. 8.

The core idea of SPP is to partition the feature map into multiple pooling windows and perform pooling operations, generating a fixed-dimensional representation. The detailed process is as follows:

1. Feature Map Partitioning: For the l-th level of the pyramid partitioning on the feature map F, the window size is defined as:

$$w_l = ([\frac{H}{n_l}], [\frac{W}{n_l}]) \tag{15}$$

Where n_l is the number of grids at the l-th level of partitioning, and H and W are the height and width of the feature map, respectively.

2. Pooling Operation: The feature values within each partitioned window are aggregated using a pooling operation to produce a scalar. Assuming the window position is (I, j), its pooled value is calculated as:

$$v^l_{i,j} = pool(F(x,y)|x \in [i \cdot w^h_l, (i+1)\cdot w^h_l), y \in [j \cdot w^w_l, (j+1)\cdot w^w_l)) \tag{16}$$

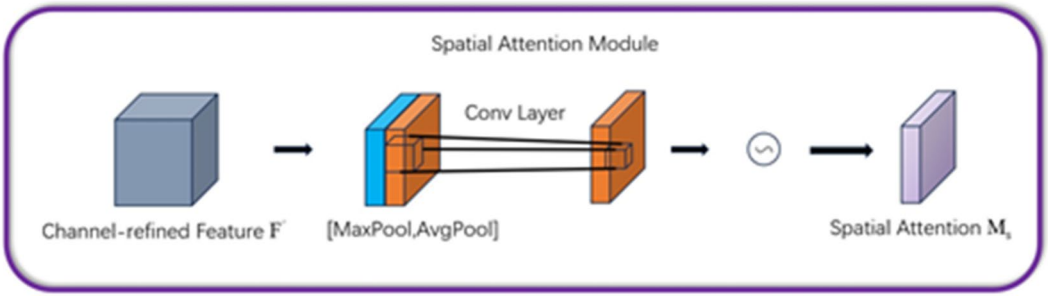

Fig. 7. Spatial attention module structure.

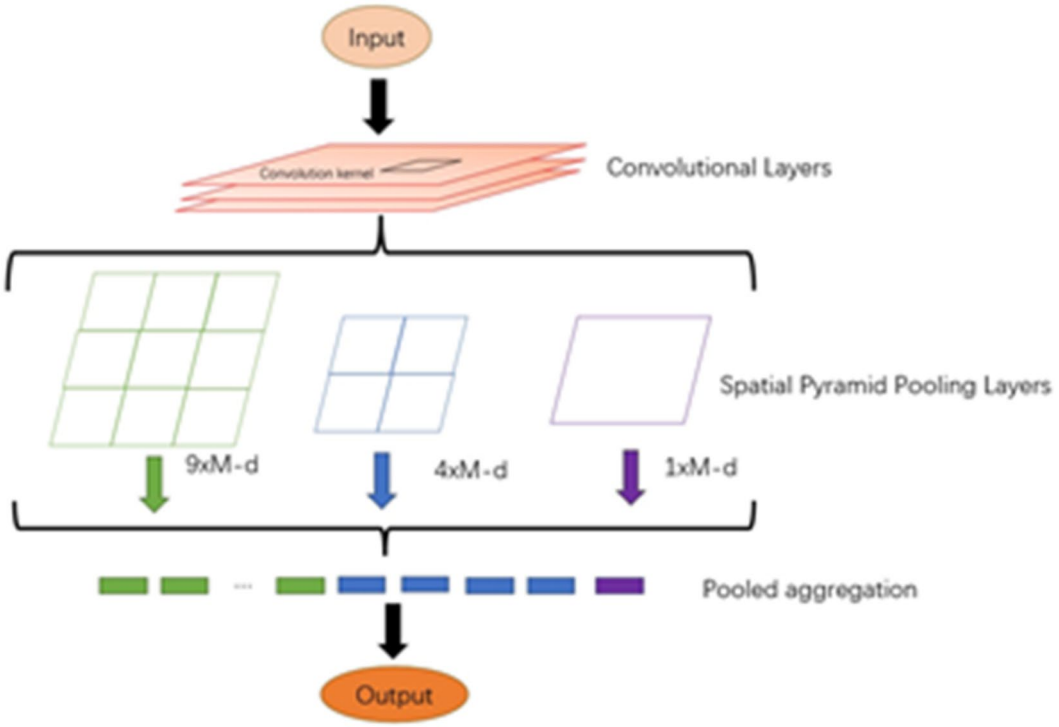

Fig. 8. Spatial pyramid pooling structure.

Fig. 9. Overall URL structure.

Where w_l^h and w_l^w represent the height and width of the window, respectively, and pool denotes the pooling function.

3. Flatten and concatenate the pooling results from all levels to form the final fixed-length representation:

$$f_{SPP} = [v^1; v^2; \ldots; v^l] \tag{17}$$

Where, v^l is the pooling result at the l-th level, with dimensions n_l^2.

Methodology

Data preprocessing

A URL is a standardized format for finding web resources and consists of components such as protocol, host, port, path, query parameters, and fragment identifiers. These components are built sequentially to form the complete resource address. Figure 9 illustrates the structure of the URL. We propose a preprocessing method for the input, where the protocol fields http and https are removed, and tokens [CLS] and [SEP] are added to the beginning and end of the domain name, respectively. This helps the CSPPC-BiLSTM model better capture the sequential information between each character in the URL after being processed through the CharEmbedding layer. The format of the pretreatment is shown in Fig. 10.

Model structure

A URL inherently has a highly sequential structure, with complex interdependencies between its characters. CSPPC-BiLSTM is an ideal model for capturing and processing such intricate sequential relationships. First,

[CLS] r77ty654.webcindario.com/a7a/detaiinfo.php[SEP]
[CLS] govuk-applicants.com[SEP]
[CLS] lumtics.com/loai/thoi-trang-nam?cs=124[SEP]
[CLS] royalbutler.co.uk/unacceptable-things-in-canada[SEP]
[CLS] www.eaglepi.com/tr/about-us/testimonials[SEP]
[CLS] www.pcsozluk.com/tag/wanacrypt0r-2-0[SEP]
[CLS] www.brewbros.ie/shop[SEP]
[CLS] www.conilmiologo.it[SEP]

Fig. 10. URL after preprocessing.

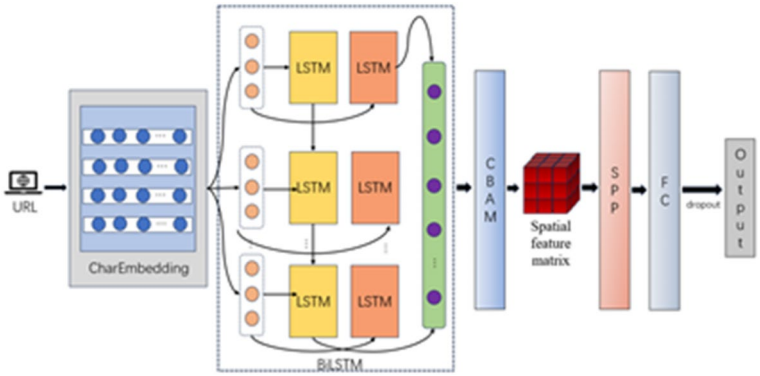

Fig. 11. Structure of the CSPPC-BiLSTM model.

the URL is passed through a character embedding layer to capture morphological features at the character level. Then, the BiLSTM extracts sequential features from the URL. To enhance feature representation, CBAM's Channel Attention Module (CAM) and Spatial Attention Module (SAM) transform these character-level sequential features into spatial sequence features, constructing a spatial feature matrix that represents the URL. Next, the SPP module applies multi-scale pooling and concatenation across different pooling window sizes in the spatial dimension. Finally, a FC layer, combined with dropout, maps the processed features into the classification space, enabling the detection and classification of malicious URLs. The structure of the CSPPC-BiLSTM model is shown in Fig. 11, with the detailed computational process as follows:

1. The input sequence $\mathrm{x} \in R^{B \times L}$ is passed through the CharEmbedding layer, transforming it into a low-dimensional vector representation $\mathrm{x_e}$:

$$\mathrm{x_e} = C\mathrm{har}E\mathrm{mbedding}(x) \tag{18}$$

B represents the batch size, and L represents the sequence length.

2. The sequence embedded by CharEmbedding is processed through BiLSTM for feature extraction:

$$h, (h_n, c_n) = BiLSTM(x_e) \tag{19}$$

h represents the output of the BiLSTM, capturing the features at each time step t in the sequence;$h_n$and c_n denote the final hidden state and cell state, respectively, which store the global information.

3. CBAM performs weighted aggregation of the time-step features extracted by BiLSTM and transforms the extracted features into a spatial feature matrix:

$$A = Soft\mathrm{max}(hW_a) \tag{20}$$

$$Spatial\ \ feature\ matrix = \sum_{t=1}^{L} \alpha_t \cdot h_t \tag{21}$$

Here, W_a represents the learnable attention weight matrix, A denotes the importance scores of the features at each time step, with Softmax as the activation function. The attention weights are α, and the context features areh_a.

4. The spatial feature matrix output by CBAM is processed through the Spatial Pyramid Pooling (SPP) module for multi-scale pooling and concatenation in the spatial dimension:

$$h'_a = Spatial\ feature\ matrix.unsqueeze(2).unsqueeze(3) \tag{22}$$

$$f_p = Concat(Spatial\ Pyramid\ Pooling(h'_a, p_i)) \tag{23}$$

h'_a is the four-dimensional tensor obtained by expanding h_a and p_i represents multiple scales.

5. Finally,f_p is passed into the fully connected layer fc, combined with dropout, to generate the final classification result:

$$y = Liner[Dropout(f_p)] \tag{24}$$

Experimental results and analysis

In this chapter, we introduce the datasets used to evaluate the performance of the CSPPC-BiLSTM model, the deep learning models selected for comparison, the experimental setup, the environment, and the evaluation metrics. Finally, we analyze the experimental results.

Datasets

To assess the performance of the proposed model, we utilized two datasets: the balanced Grambedding dataset[31] (https://web.cs.hacettepe.edu.tr/~selman/grambeddings-dataset/) and the imbalanced Mendeley AK Singh dataset[32] (https://data.mendeley.com/datasets/gdx3pkwp47/2). In both datasets, benign URLs are labeled as "legit" and malicious URLs are labeled as "malicious." The balanced Grambedding dataset contains a total of 960,083 samples, with 480,088 legit URLs and 479,995 malicious URLs. On the other hand, the imbalanced Mendeley AK Singh dataset comprises 1,530,708 samples, including 1,495,542 legit URLs and 35,166 malicious URLs. The frequency distribution of URLs in both datasets is illustrated in Fig. 12 using bar charts.

Comparison methods

We compared the CSPPC-BiLSTM model with six advanced character-level deep learning models, including:

CharBiGRU: This model uses a character-level BiGRU (Bidirectional Gated Recurrent Unit, BiGRU) network, with a dropout layer and a FC layer following the BiGRU layer.

CharBiLSTM: A model based on character-level BiLSTM network.

CharGRU: A character-level GRU (Gated Recurrent Unit, GRU) network.

CharLSTM: A character-level LSTM (Long Short-Term Memory, LSTM) network.

CharCNN: A character-level CNN that uses convolutional kernels of sizes 3, 4, and 5, followed by a dropout layer and a fully connected layer.

CharCNNBiLSTM: This model combines a character-level CNN with a BiLSTM network. Features are extracted from the embedding layer using the CNN, and then passed to the BiLSTM for further processing.

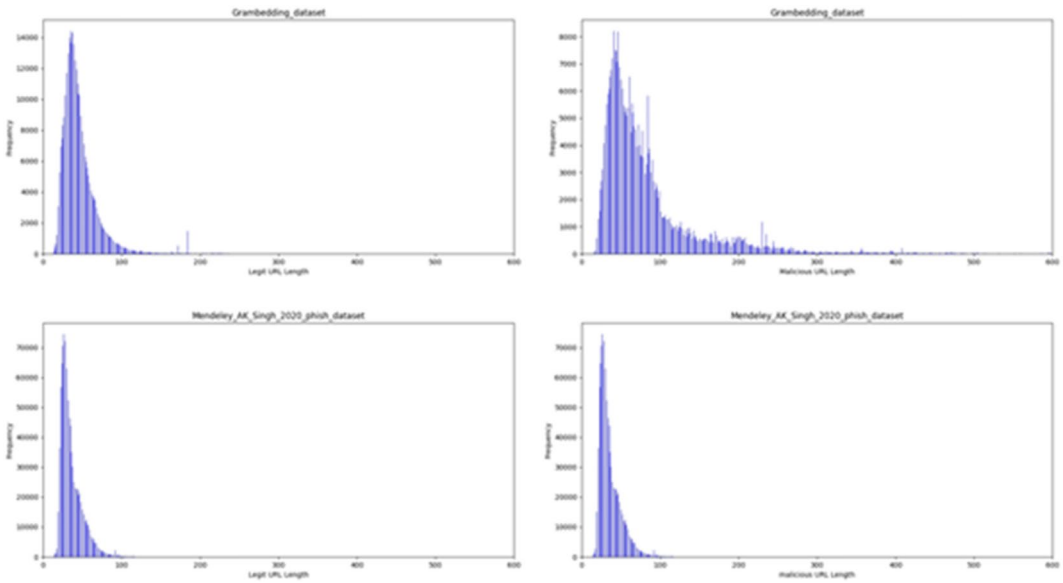

Fig. 12. Bar Chart of URL Frequence.

Experimental setup and results

We used the PyTorch framework (version cuda = 12.4) with Python 3.8 for all experiments. These were conducted on a server equipped with a 12th Gen Intel(R) Core (TM) i7-12700KF CPU and an NVIDIA GeForce RTX 4070 GPU.

Comparative experiments and results

For all models, the number of epochs, batch size, and learning rate were set to 20, 256, and 0.001, respectively. Since the experiment involves balanced data sets and unbalanced data sets, we also quoted 12 evaluation indicators to comprehensively evaluate the performance of CSPPC-BiLSTM in facing balanced data and unbalanced data from multiple perspectives. The following is a detailed introduction and significance of each evaluation indicator. The experimental results are shown in Table 1. The ROC curves and confusion matrices of the experimental results in the balanced data set Grambedding Dataset are shown in Figs. 13 and 14. The ROC curves and confusion matrices of the experimental results in the unbalanced data set Mendeley Phish Dataset are shown in Figs. 15 and 16.

Accuracy: $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$ Accuracy represents the proportion of correctly predicted samples to the total number of samples. Reflects overall prediction accuracy, but may be inaccurate when classes are imbalanced.

Precision: $Precision = \frac{TP}{TP+FP}$ Precision represents the proportion of true positives among the samples predicted as positive. Reflects the reliability of the model in predicting positive classes. High accuracy means fewer false positives.

Recall: $Recall = \frac{TP}{TP+FN}$ Recall represents the proportion of actual positive samples correctly predicted as positive. Reflecting the model's ability to capture positive classes, high recall means fewer false negatives.

F1Score: $F1\ Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + \mathrm{Re}\,calL}$ The F1 score is the harmonic mean of precision and recall, providing a comprehensive evaluation of model performance. Reflects a balance between precision and recall and is suitable for class imbalance situations.

Micro F1 Score (F1_mic): $F1\_micro = \frac{2 \cdot TP}{2 \cdot TP+FP+FN}$ Used to assess data with class imbalance. Reflects overall performance, suitable for class-balanced datasets.

Weighted F1 Score (F1_wted): $F1\_wted = \sum_{i=1}^{n} \frac{s_i}{S} \cdot F1_i$ Suitable for scenarios involving class imbalance. It reflects that it can better reflect the model performance when the class is imbalanced.

False Discovery Rate (FDR): $FDR = \frac{FP}{TP+FP}$ Represents the proportion of false positives among the samples predicted as positive. Reflecting the proportion of false positives, a low FDR means the model is more reliable in predicting positive classes.

False Negative Rate (FNR): $FNR = \frac{FN}{TP+FN}$ Represents the proportion of actual positive samples misclassified as negative. Reflecting the proportion of false negatives, low FNR means that the model has a strong ability to capture positive classes.

False Positive Rate (FPR): $FPR = \frac{FP}{FP+TN}$ Represents the proportion of actual negative samples misclassified as positive. Reflecting the proportion of false positives, low FPR means that the model has a strong ability to identify negative classes.

Dataset	Name	Accuracy	Precision	Recall	F1Score	F1_micro	F1_wted	FDR	FNR	FPR	DE	NPV	SPC
Grambedding Dataset	CharBiGRU	97.67	97.67	97.67	97.67	97.70	97.70	2.294	2.295	2.295	95.46	97.70	97.70
	CharBiLSTM	97.54	97.55	97.54	97.54	97.53	97.53	2.461	2.468	2.468	95.12	97.53	97.53
	CharGRU	97.52	97.52	97.52	97.52	97.53	97.52	2.460	2.461	2.461	95.13	97.53	97.53
	CharLSTM	97.57	97.58	97.57	97.57	97.57	97.57	2.407	2.421	2.421	95.20	97.59	97.57
	CharCNN	96.64	96.65	96.64	96.64	97.14	97.14	2.846	2.854	2.854	94.36	97.15	97.14
	CharCNNBiLSTM	96.78	96.79	96.78	96.78	96.83	96.83	3.157	3.166	3.166	93.76	96.84	96.83
	CSPPC-BiLSTM	**99.66**	**99.66**	**99.66**	**99.66**	**99.81**	**99.81**	**0.185**	**0.185**	**0.185**	**99.62**	**99.81**	**99.81**
Mendeley phish	CharBiGRU	98.86	98.78	98.86	98.74	98.88	98.76	5.257	21.936	21.936	56.18	94.74	78.06
	CharBiLSTM	98.88	98.80	98.88	**98.75**	98.88	98.75	4.980	22.130	22.130	55.79	95.01	77.86
	CharGRU	98.85	98.75	98.85	98.73	98.85	98.74	7.067	**21.240**	**21.240**	**57.60**	92.93	**78.75**
	CharLSTM	98.82	98.73	98.82	98.69	98.83	98.70	6.426	22.432	22.432	55.21	93.57	77.56
	CharCNN	98.78	98.67	98.78	98.63	98.85	98.70	4.916	23.078	23.078	53.89	95.08	76.92
	CharCNNBiLSTM	98.75	98.68	98.75	**98.75**	98.80	98.64	5.100	24.228	24.228	51.59	94.89	75.77
	CSPPC-BiLSTM	**98.89**	**98.82**	**98.89**	**98.75**	**98.90**	**98.77**	**4.705**	21.971	21.971	56.10	**95.29**	78.02

Table 1. Presents the results of malicious URL detection. Metrics with the best performance are highlighted in bold, and all performance metrics are expressed as percentages.

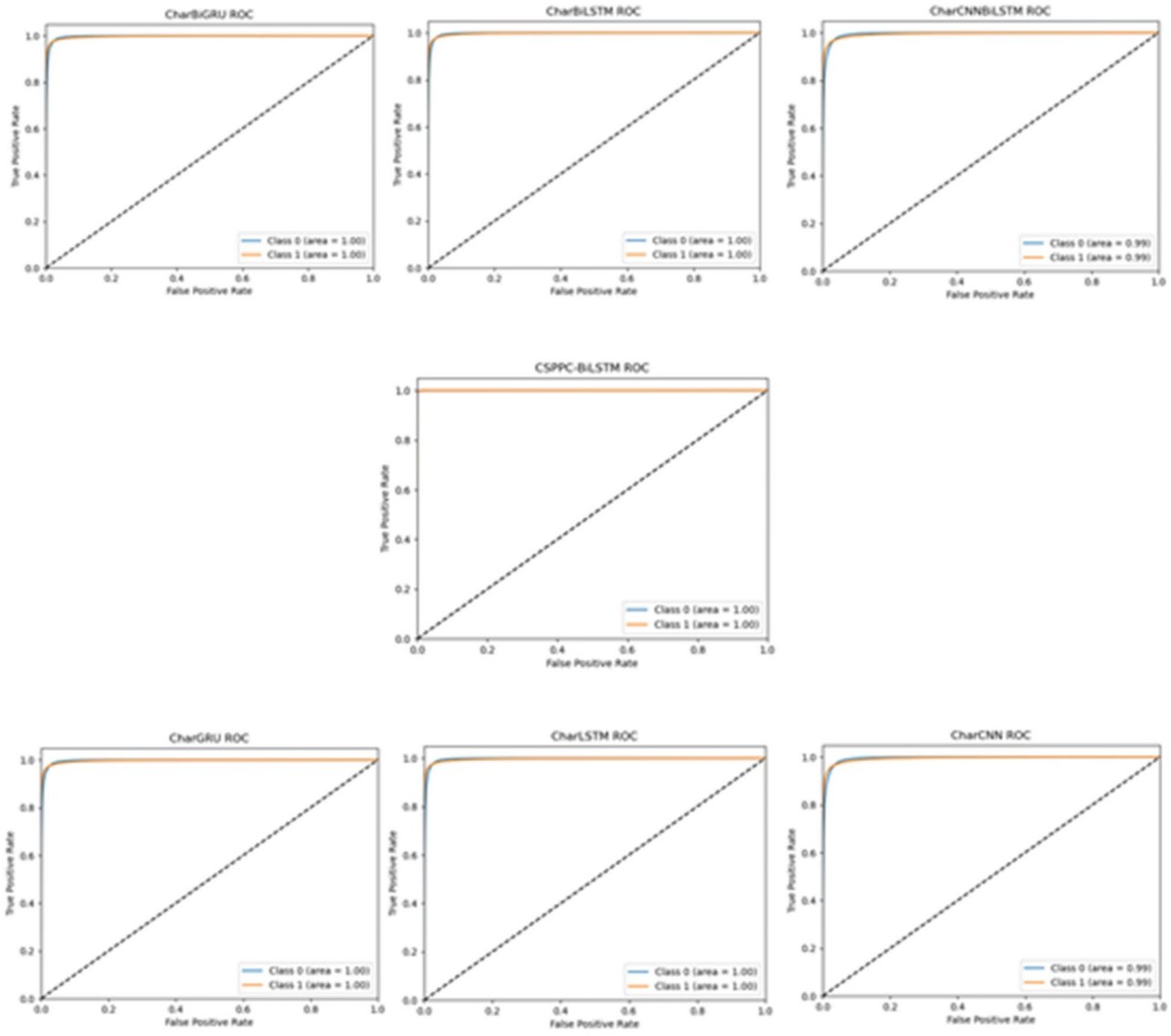

Fig. 13. ROC Curve of models on the Grambedding dataset.

Diagnostic Efficiency (DE): $DE = \frac{TP+TN}{TP+FP++TN+FN}$ Represents the overall accuracy rate of the model in classifying positive and negative samples. Reflecting the overall error rate, high DE means that the overall performance of the model is better.

Negative Predictive Value (NPV): $NPV = \frac{TN}{TN+FN}$ Represents the proportion of true negatives among the samples predicted as negative. Reflecting the reliability of the model in predicting negative classes, high NPV means fewer false positives.

Specificity (SPC): $Specificity = \frac{TN}{FP+TN}$ Represents the proportion of actual negative samples correctly predicted as negative. Reflecting the model's ability to identify negative classes, high SPC means fewer false positives.

Figures 13 and 14 illustrate the ROC curve and confusion matrix of the CSPPC-BiLSTM model compared to other models on the Grambedding Dataset. Similarly, Figs. 15 and 16 present the ROC curve and confusion matrix of the CSPPC-BiLSTM model compared to other models on the Mendeley Phish dataset.

The results indicate that CSPPC-BiLSTM, integrating CBAM and SPP, successfully transforms character sequence features into spatial sequence features, demonstrating effectiveness in malicious URL detection. The estimated experimental values are presented in Table 1, while Figs. 13, 14 and 15, and 16 provide a more intuitive visualization of the results.

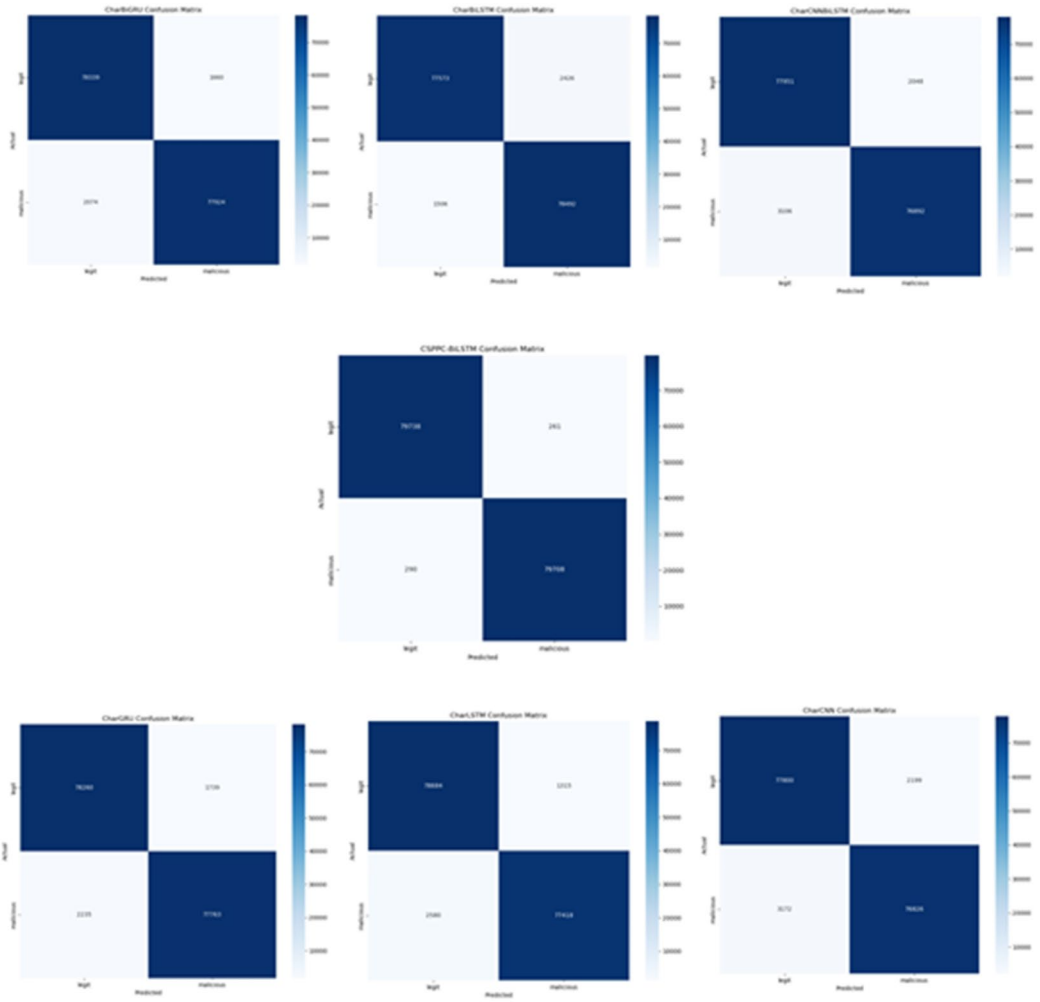

Fig. 14. Confusion matrix of models on the Grambedding dataset.

On the balanced Grambedding dataset, CSPPC-BiLSTM showed strong and stable performance. Compared with six commonly used character-level neural models, it achieved a 2–3% improvement on the four most commonly used evaluation metrics. It is worth noting that FDR and FNR reached 0.185, indicating that the model has strong capture and recognition capabilities for positive samples. FPR and FPC reached 0.185 and 99.81 respectively, indicating that the model has excellent detection capabilities for negative samples. Other evaluation metrics also reflect the relatively excellent performance of CSPPC-BiLSTM. As can be seen from Fig. 14, for malicious URL detection by CSPPC-BiLSTM, the proportion of false detections is significantly reduced compared with other models (the white part of the confusion matrix).

For the unbalanced Mendeley AK Singh dataset, Table 1; Figs. 15 and 16 show that CSPPC-BiLSTM outperforms other character-level neural network models in most evaluation indicators, except that its performance is slightly lower than CharBiGRU in FNR, FPR, DE, and SPC. As can be seen from Fig. 16, when faced with data imbalance, CSPPC-BiLSTM is more likely to misjudge originally malicious URLs as legitimate URLs compared to CharGRU, and more likely to misjudge originally legitimate URLs as malicious URLs compared to CharCNNBiLSTM. CSPPC-BiLSTM has a certain misjudgment rate when the data is unbalanced. Despite this, compared with other character RNNs and CNN-based models, CSPPC-BiLSTM still shows quite good performance, and the detection effect of malicious URLs has been significantly improved.

It can be concluded that CSPPC-BiLSTM has good malicious URL detection accuracy when faced with balanced data. When faced with unbalanced data, CSPPC-BiLSTM still performs well, but is slightly insufficient in some indicators. This is also the focus of my future research.

Ablation experiments and results

In order to show the combination of CBAM and SPP more comprehensively, it is a bridge connecting character-level sequence features and spatial sequence features, which is of great significance to CSPPC-BiLSTM. We conducted ablation experiments using CSPPC-BiLSTM and CC-BiLSTM (removing SPP), CSPP-BiLSTM (removing CBAM) models to prove the significance of combining CBAM with SPP. The results of the ablation

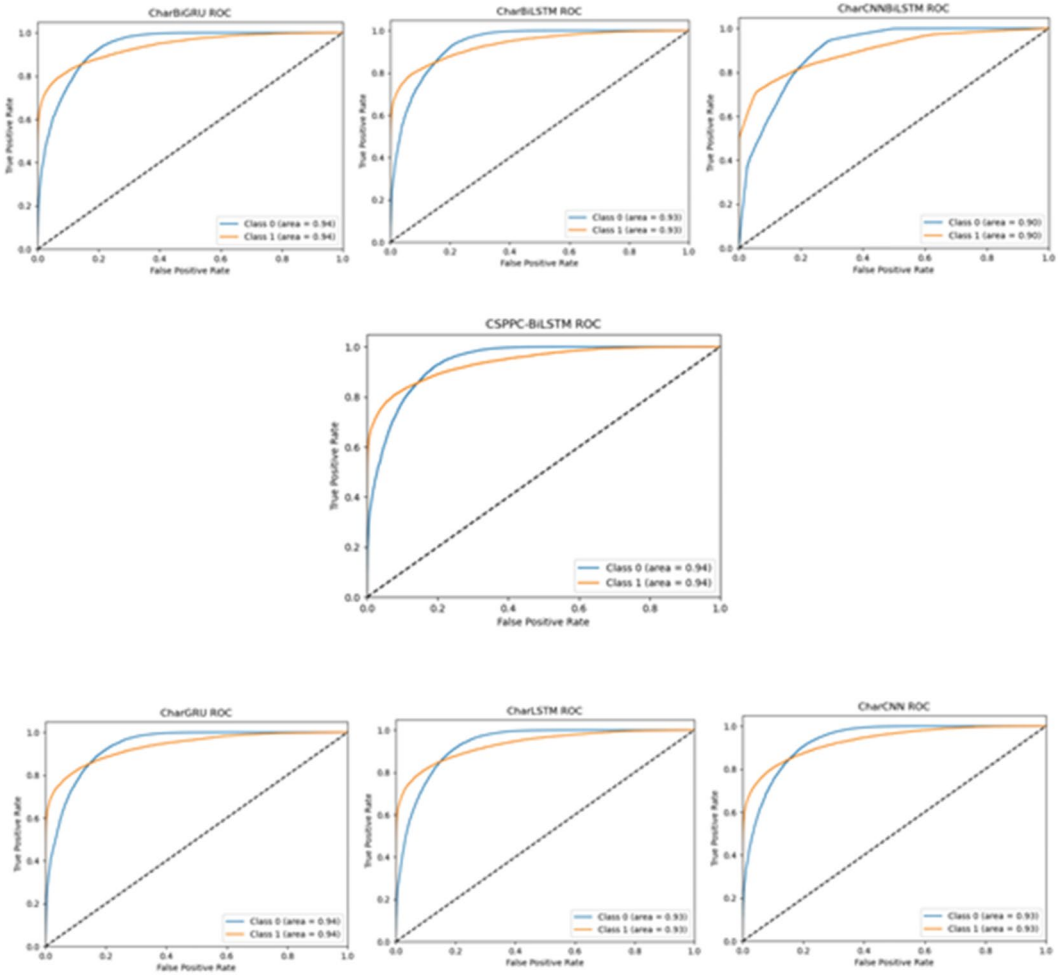

Fig. 15. ROC curve of models on the Mendeley phish dataset.

experiment are shown in Table 2; Figs. 17, 18, 19 and 20. The evaluation indicators used for experimental result analysis are consistent with the 4.3.1 Comparative Experiment Chapter. Figures 17 and 18 correspond to the balanced data set Grambedding Dataset, and Figs. 19 and 20 correspond to the unbalanced data set Mendeley Phish.

As can be seen from Table 2; Figs. 17, 18, 19 and 20, when faced with balanced data, CSPPC-BiLSTM has a significantly lower number of misjudged URLs than CC-BiLSTM and CSPP-BiLSTM. CSPPC-BiLSTM misjudged 551 data items, while CC-BiLSTM and CSPP-BiLSTM misjudged 3457 and 3932 data items, respectively. The results prove the effectiveness of combining CBAM and SPP in processing balanced data. Even in the face of unbalanced data, CSPPC-BiLSTM is still better than the other two models in some aspects. The number of URL misjudgments has been decreasing relative to CC-BiLSTM and CSPP-BiLSTM. Compared with CC-BiLSTM, CSPP-BiLSTM seems to be more inclined to misjudge originally malicious URLs as legitimate URLs. This shows that using SPP alone without CBAM will directly affect the performance of the model in some aspects, because SPP directly processes the features extracted by BiLSTM without an intermediate mechanism to convert character-level sequence features into spatial sequence representations. On the other hand, although the channel attention mechanism using only CBAM can effectively capture character-level sequence features and the spatial attention mechanism can convert character sequence features into spatial sequence features, it lacks SPP for further spatial pooling operations, which will lead to a decrease in the performance of the model. However, integrating SPP on top of CBAM can serve as a bridge to further refine the spatial sequence features extracted by CBAM spatial attention. This study innovatively integrates CBAM and SPP, so that CSPPC-BiLSTM enhances the ability to learn key malicious URL features in the spatial domain, ultimately improving detection accuracy.

Conclusions

In this work, we introduce the CSPPC-BiLSTM model, designed to capture the sequential characteristics of malicious URLs on the web. The model first employs a character embedding layer to convert discrete inputs into continuous representations, followed by a BiLSTM to extract inter-character dependencies within URLs. CSPPC-

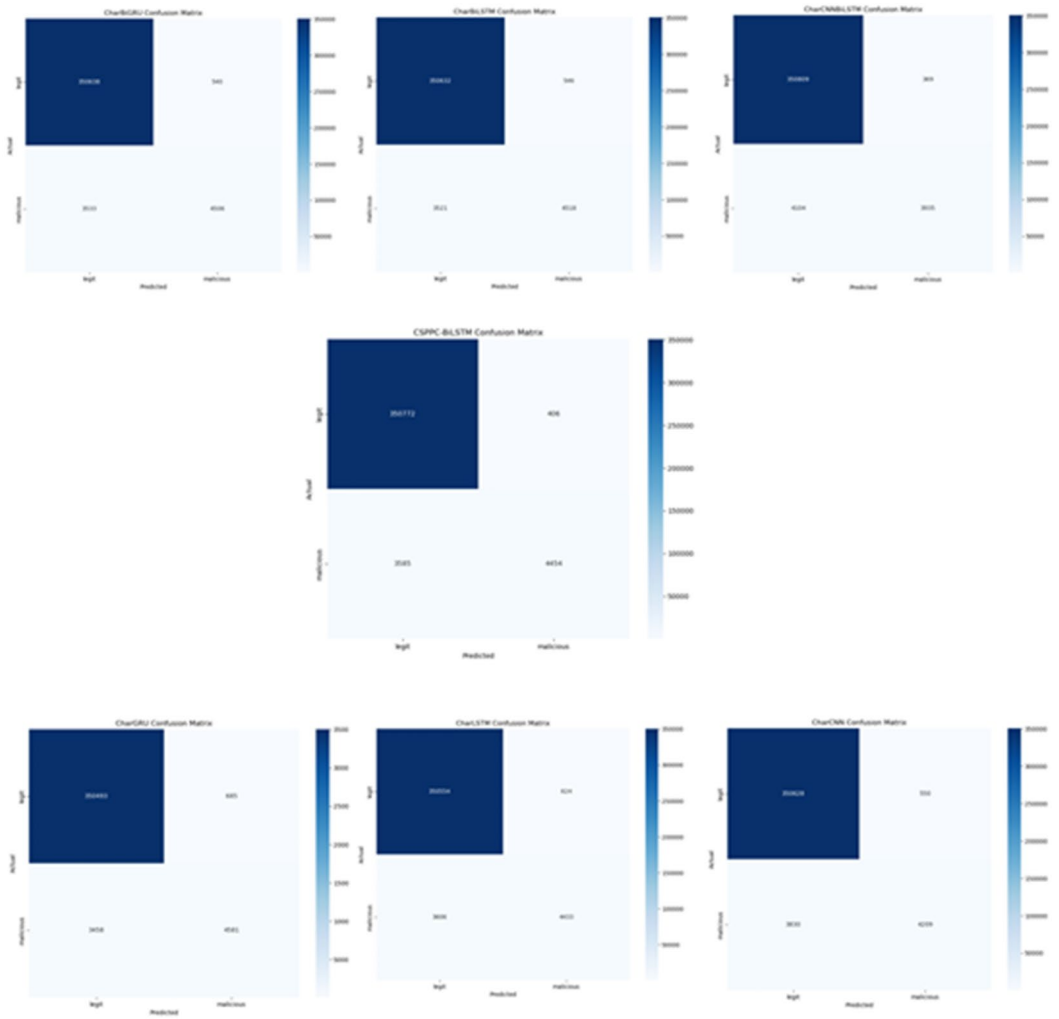

Fig. 16. Confusion matrix of models on the Mendeley Phish dataset.

Dataset	Name	Accuracy	Precision	Recall	F1Score	F1_micro	F1_wted	FDR	FNR	FPR	DE	NPV	SPC
Grambedding Dataset	CSPP-BiLSTM	97.54	97.55	97.54	97.54	97.53	97.53	2.461	2.468	2.468	95.12	97.53	97.53
	CC-BiLSTM	97.84	97.84	97.84	97.91	97.91	97.91	2.079	2.083	2.083	95.87	97.92	97.91
	CSPPC-BiLSTM	**99.66**	**99.66**	**99.66**	**99.66**	**99.81**	**99.81**	**0.185**	**0.185**	**0.185**	**99.62**	**99.81**	**99.81**
Mendeley phish	CSPP-BiLSTM	98.81	98.71	98.81	98.70	98.84	98.72	7.329	**21.474**	**21.474**	57.13	92.67	**78.52**
	CC-BiLSTM	98.82	98.72	98.82	98.70	98.84	98.72	6.834	21.714	21.714	**56.65**	93.16	78.28
	CSPPC-BiLSTM	**98.89**	**98.82**	**98.89**	**98.75**	**98.90**	**98.77**	**4.705**	21.971	21.971	56.10	**95.29**	78.02

Table 2. Results of ablation experiment malicious URL detection. Metrics with the best performance are highlighted in bold, and all performance metrics are expressed as percentages.

BiLSTM integrates two powerful modules, CBAM and SPP. Its key innovation lies in leveraging CBAM's channel and spatial attention mechanisms to refine the sequential features extracted by BiLSTM, which are then further aggregated through spatial pooling with SPP. Finally, classification is performed via a fully connected layer. Through comparative and ablation experiments on both balanced and imbalanced datasets, CSPPC-BiLSTM demonstrates superior accuracy and robustness compared to conventional malicious URL detection models, highlighting the effectiveness of combining CBAM and SPP. This study represents an integrated innovation at the feature extraction level, significantly enhancing phishing URL detection capabilities. It effectively addresses key challenges in feature extraction, attention modeling, and spatial representation learning, thereby improving both detection accuracy and robustness.

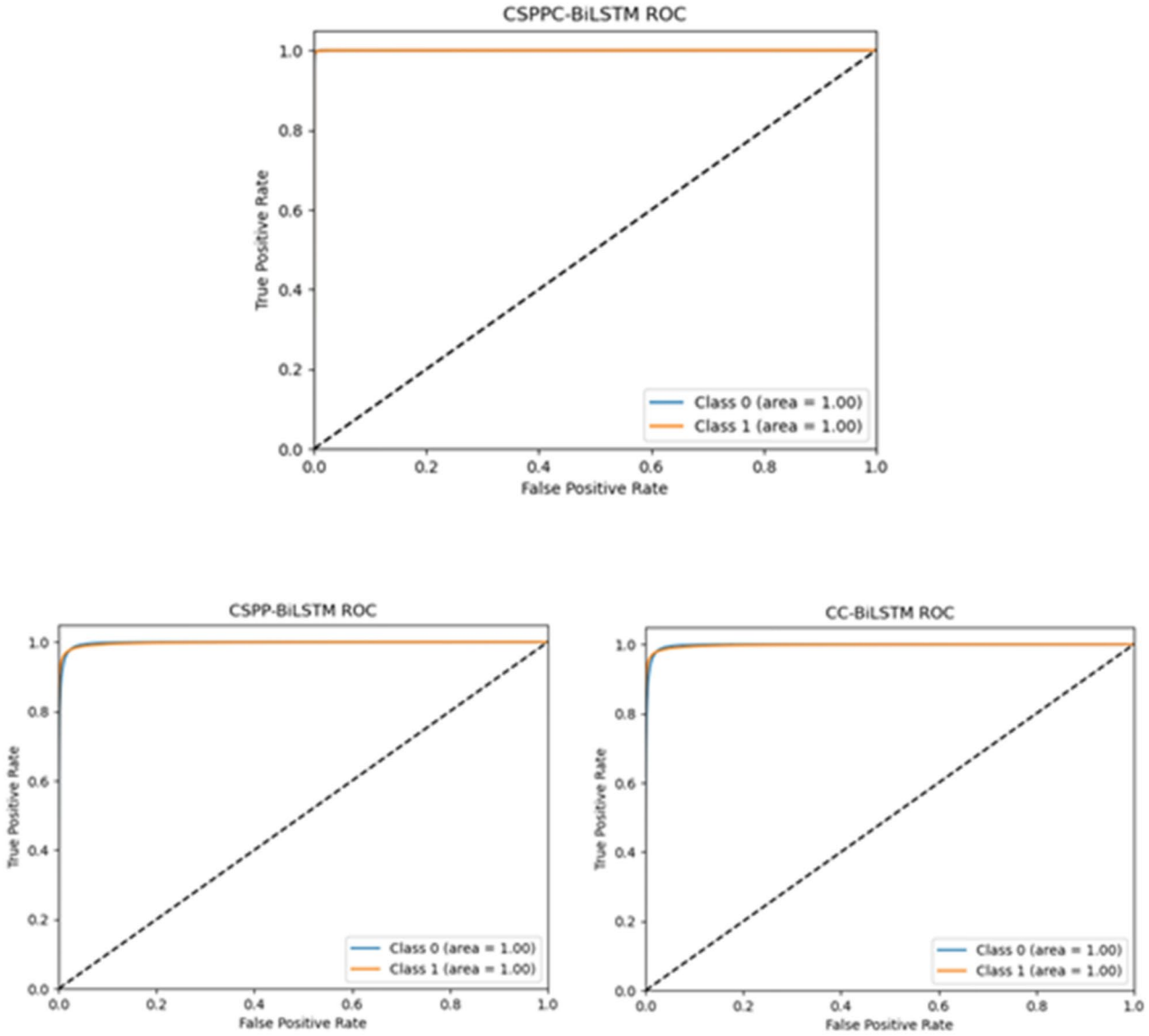

Fig. 17. Ablation experiment ROC curve on the Grambedding dataset.

In future work, we aim to further improve the model's detection efficiency and accuracy when dealing with imbalanced data. We plan to explore more advanced attention mechanisms and further optimize the feature extraction module to maintain high detection performance in the face of obstacles such as data imbalance.

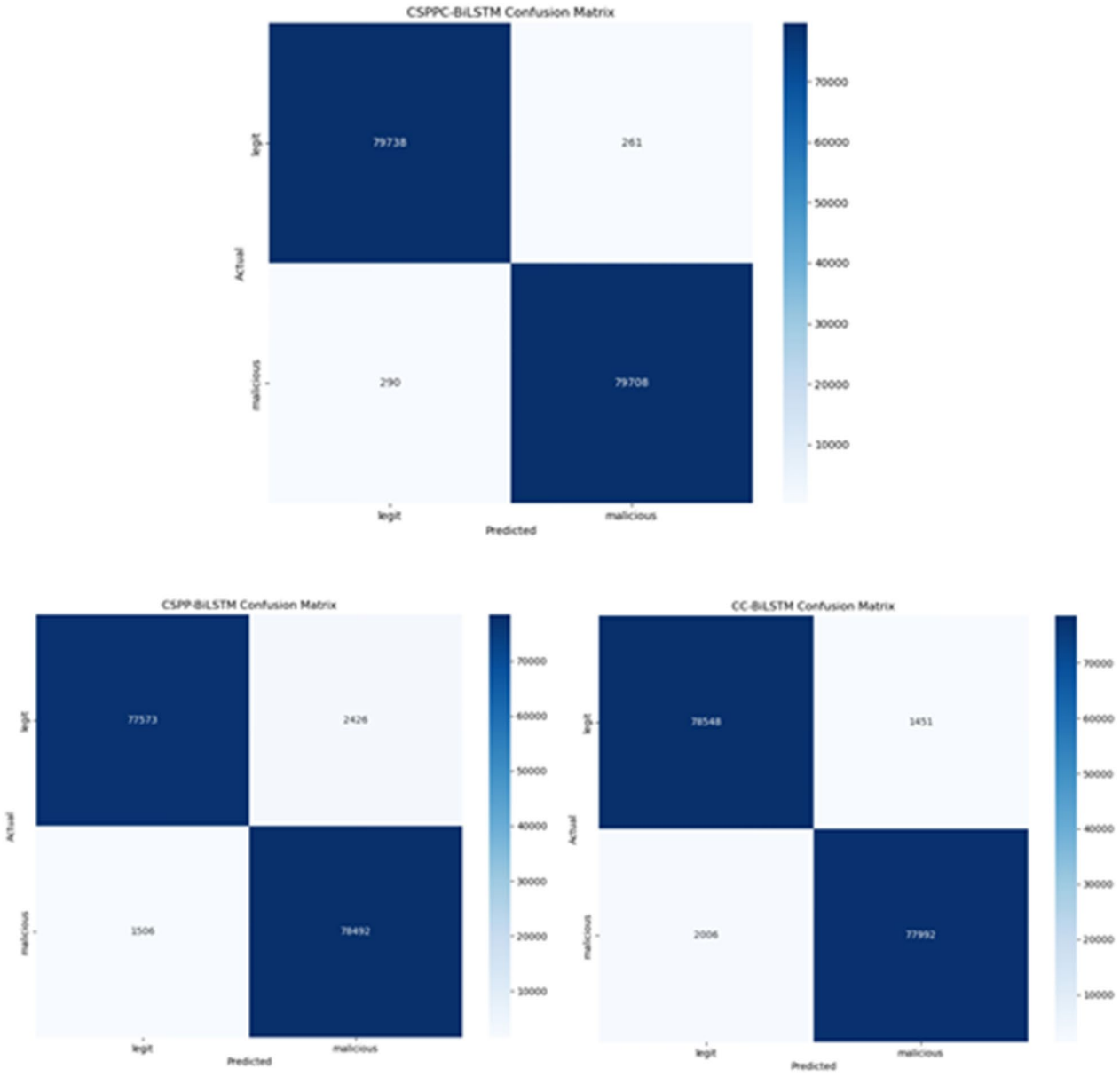

Fig. 18. Ablation experiment confusion matrix on the Grambedding dataset.

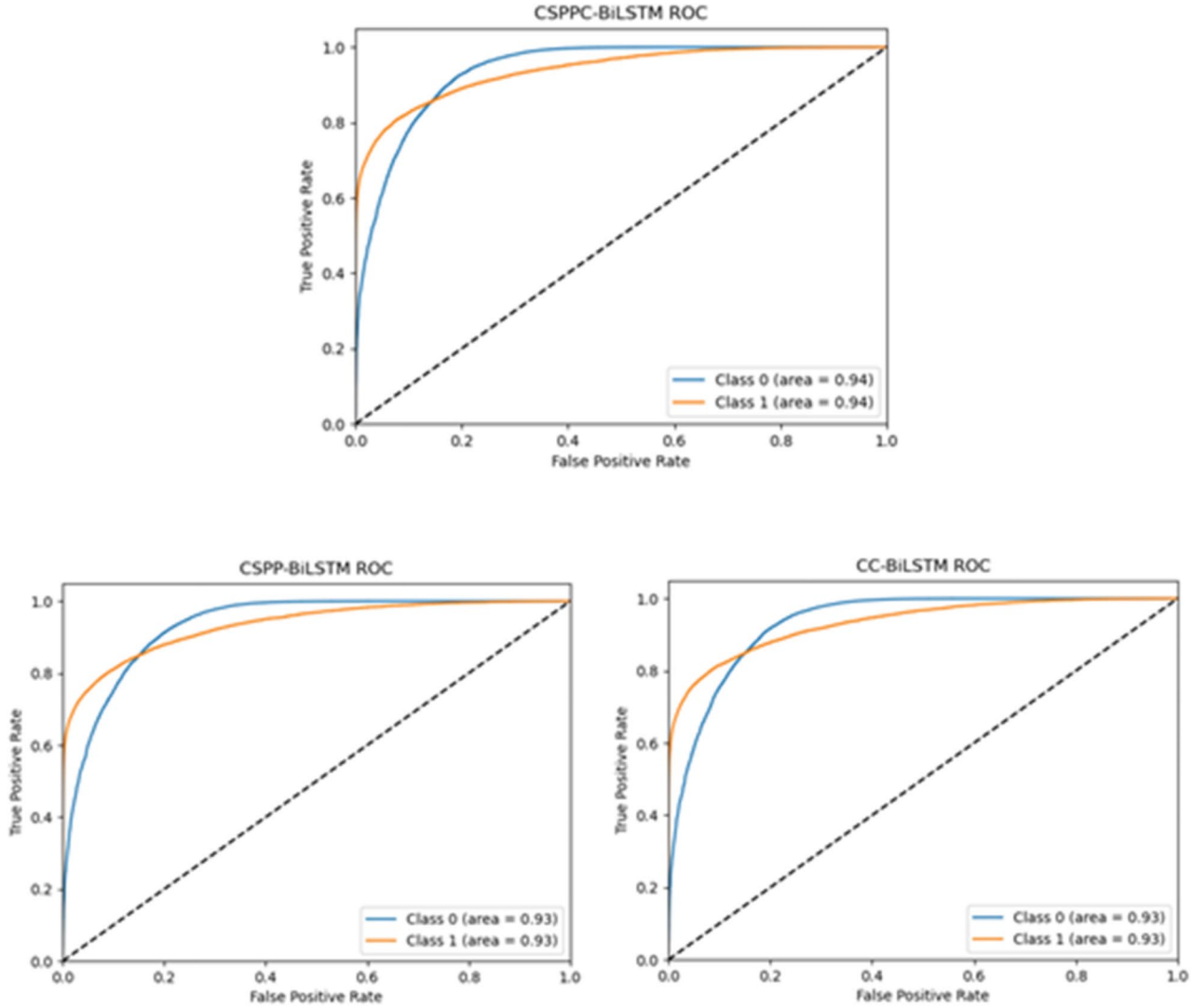

Fig. 19. Ablation experiment ROC curve on the Mendeley Phish dataset.

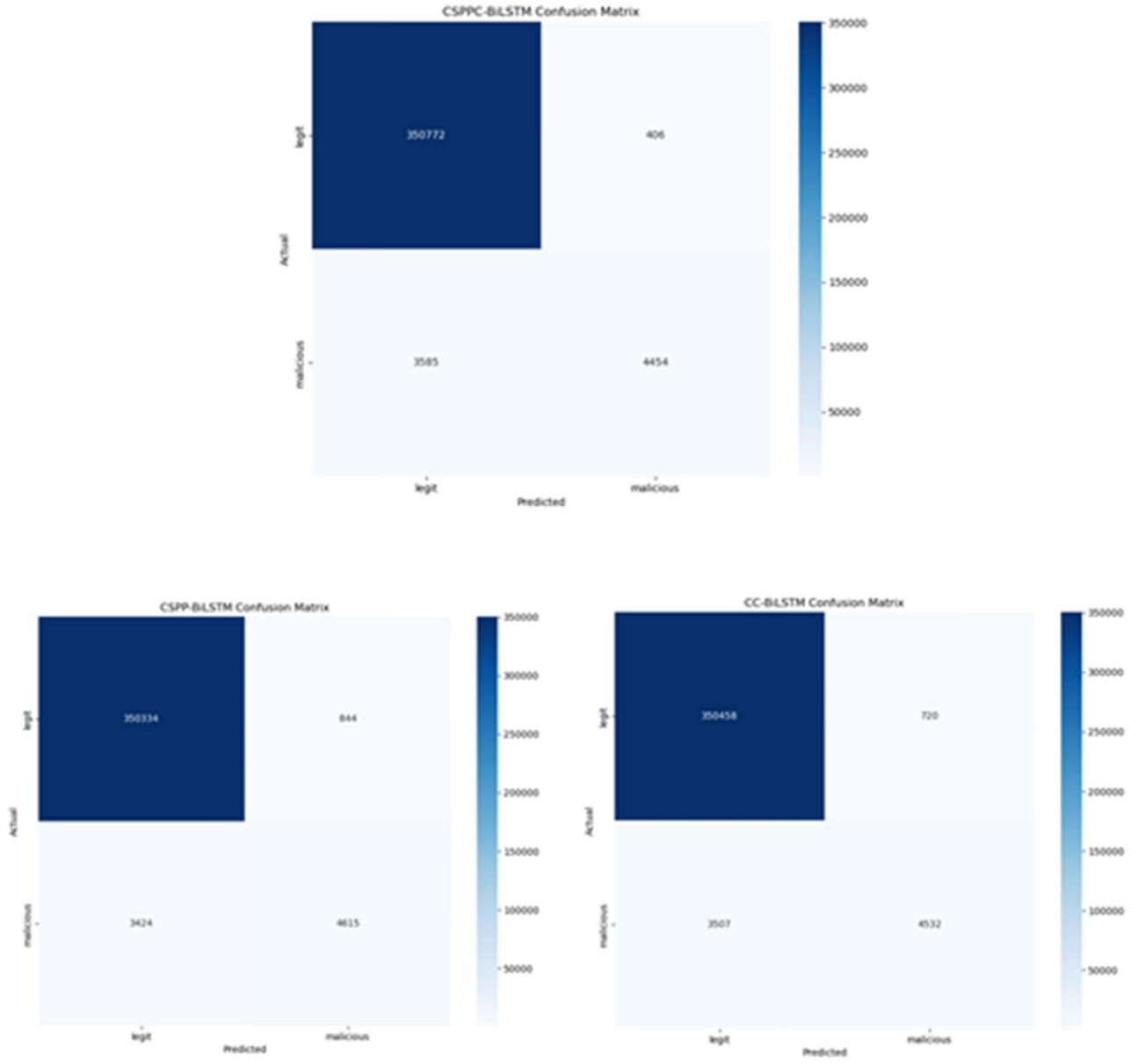

Fig. 20. Ablation experiment confusion matrix on the Mendeley Phish dataset.

Data availability

To assess the performance of the proposed model, we utilized two datasets: the balanced Grambedding dataset[31] (https://web.cs.hacettepe.edu.tr/~selman/grambeddings-dataset/) and the imbalanced Mendeley AK Singh dat aset[32] (https://data.mendeley.com/datasets/gdx3pkwp47/2). In both datasets, benign URLs are labeled as "legit" and malicious URLs are labeled as "malicious." The balanced Grambedding dataset contains a total of 960,083 samples, with 480,088 legit URLs and 479,995 malicious URLs. On the other hand, the imbalanced Mendeley AK Singh dataset comprises 1,530,708 samples, including 1,495,542 legit URLs and 35,166 malicious URLs.

Received: 26 November 2024; Accepted: 18 February 2025
Published online: 24 February 2025

References

1. Atrees, M., Ahmad, A. & Alghanim, F. *Enhancing Detection of Malicious URLs Using Boosting and Lexical Features* 31 (Intelligent Automation & Soft Computing, 2022). 3.
2. Vanhoenshoven, F. et al. Detecting malicious URLs using machine learning techniques. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 1–8 (2016).

3. Wei, W. et al. Accurate and fast URL phishing detector: A convolutional neural network approach. *Comput. Netw.* **178**, 107275 (2020).
4. Afzal, S. et al. Urldeepdetect: A deep learning approach for detecting malicious urls using semantic vector models. *J. Netw. Syst. Manag.* **29**, 1–27 (2021).
5. Wang, H. et al. Bidirectional LSTM malicious webpages detection algorithm based on convolutional neural network and independent recurrent neural network. *Appl. Intell.* **49**, 3016–3026 (2019).
6. Le, H. et al. URLNet: Learning a URL representation with deep learning for malicious URL detection. *ArXiv Preprint* (2018). arXiv:1802.03162.
7. Liang, Y. et al. Robust detection of malicious urls with self-paced wide & deep learning. *IEEE Trans. Depend. Secur. Comput.* **19** (2), 717–730 (2021).
8. Wu, T. et al. Malicious url detection model based on bidirectional gated recurrent unit and attention mechanism. *Appl. Sci.* **12** (23), 12367 (2022).
9. Nanda, M. & Goel, S. URL based phishing attack detection using BiLSTM-gated highway attention block convolutional neural network. *Multimed. Tools Appl.*, 1–31. (2024).
10. Liu, R. et al. Malicious URL detection via pretrained Language model guided Multi-Level feature attention network. arXiv preprint arXiv:2311.12372 (2023).
11. Mahdaouy, A. E. et al. DomURLs_BERT: Pre-trained BERT-based model for malicious domains and URLs detection and classification. arXiv preprint arXiv:2409.09143 (2024).
12. Nowroozi, E., Mohammadi, M. & Conti, M. An adversarial attack analysis on malicious advertisement URL detection framework. *IEEE Trans. Netw. Serv. Manag.* **20** (2), 1332–1344 (2022).
13. Zhu, E. et al. CCBLA: a lightweight phishing detection model based on CNN, BiLSTM, and attention mechanism. *Cogn. Comput.* **15** (4), 1320–1333 (2023).
14. Tsai, Y. D. et al. Toward more generalized malicious url detection models. *Proc. AAAI Conf. Artif. Intell.* **38** (19), 21628–21636 (2024).
15. Jeeva, S. C. & Rajsingh, E. B. Intelligent phishing url detection using association rule mining. *Hum.-Cent. Comput. Inform. Sci.* **6**, 1–19 (2016).
16. Lee[1], O. V. et al. A malicious urls detection system using optimization and machine learning classifiers. *Indones. J. Electr. Eng. Comput. Sci.* **17** (3), 1210–1214 (2020).
17. AlEroud, A. & Karabatis, G. Bypassing detection of URL-based phishing attacks using generative adversarial deep neural networks. In *Proceedings of the Sixth International Workshop on Security and Privacy Analytics*, 53–60 (2020).
18. Karajgar, M. D. et al. Comparison of machine learning models for identifying malicious URLs. In *2024 IEEE International Conference on Information Technology, Electronics and Intelligent Communication Systems (ICITEICS)*. IEEE, 1–5. (2024).
19. Ozcan, A. et al. A hybrid DNN–LSTM model for detecting phishing urls. *Neural Comput. Appl.*, 1–17 (2023).
20. Remya, S. et al. *An Effective Detection Approach for Phishing URL Using ResMLP* (IEEE Access, 2024).
21. Islam, M. S. et al. WebGuardML: Safeguarding users with malicious URL detection using machine learning. In *2023 26th International Conference on Computer and Information Technology (ICCIT)*. IEEE, 1–6. (2023).
22. Mia, M., Derakhshan, D. & Pritom, M. M. A. Can features for phishing URL detection be trusted across diverse datasets? A case study with explainable AI. In *Proceedings of the 11th International Conference on Networking, Systems, and Security*. 137–145 (2024).
23. Kaushik, P. & Rathore, S. P. S. Deep learning multi-agent model for phishing cyber-attack detection. *Int. J. Recent. Innov. Trends Comput. Commun.* **11** (9s), 680–686 (2023).
24. Su, M. Y. & Su, K. L. BERT-Based approaches to identifying malicious urls. *Sensors* **23** (20), 8499 (2023).
25. Geyik, B., Erensoy, K. & Kocyigit, E. Detection of phishing websites from urls by using classification techniques on WEKA. In *6th Int. Conf. Inventive Comput. Technol. (ICICT)*. IEEE (2021), 120–125 (2021).
26. Taofeek, A. O. Development of a novel approach to phishing detection using machine learning. *ATBU J. Sci. Technol. Educ.* **12** (2), 336–351 (2024).
27. Schuster, M. & Paliwal, K. K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **45** (11), 2673–2681 (1997).
28. Hochreiter, S. et al. Long short-term memory. Neural Computation MIT-Press, Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, 3–19. (1997).
29. Woo, S. et al. Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*. : 3–19 (2018).
30. Zhao, H. et al. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2881–2890 (2017).
31. Bozkir, A. S., Dalgic, F. C. & Aydos, M. GramBeddings: A new neural network for URL based identification of phishing web pages through n-gram embeddings. *Comput. Secur.* **124**, 102964 (2023).
32. Singh, A. K. *Dataset of Malicious and Benign Webpages* (Mendeley Data, 2020).

Acknowledgements

This paper was supported by the Hainan Province Science and Technology Special Fund (Fund No. ZDYF-2024GXJS034); the Innovation Platform for Academicians of Hainan Province (Fund No. YSPTZX202036); the Sanya Science and Technology Special Fund (Fund No. 2022KJCX30).

Author contributions

All authors participated in the manuscript review.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to K.Z., A.B. or Y.Z.