Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Deep Learning Framework for Phishing URL Detection Using Multilevel Embeddings and Attention Based Feature Fusion

Chuan-Bi Lin

cblin@cyut.edu.tw

Chaoyang University of Technology

Yi-Kai Liu

Chaoyang University of Technology

Alok Kumar Sharma

Chaoyang University of Technology

Research Article

Keywords: Phishing URL Detection, Deep Learning, Attention Mechanism, Feature Fusion, Multi-Level Embedding

Posted Date: November 19th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11263678/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

A Deep Learning Framework for Phishing URL Detection Using Multilevel Embeddings and Attention Based Feature Fusion

Chuan-Bi Lin[1, *], Yi-Kai Liu[1] and Alok Kumar Sharma[1]

[1]Department of Information and Communication Engineering

Chaoyang University of Technology, Taichung, Taiwan

Corresponding Author: *cblin@cyut.edu.tw

Abstract

Phishing attacks remain a major cybersecurity threat, while conventional blacklist- and handcrafted feature-based detection methods have limited capability to identify previously unseen and rapidly evolving phishing URLs. This study proposes PhishDetect, a representation-aware multi-branch deep learning framework that learns complementary structural and contextual characteristics directly from URLs. Unlike approaches that rely on a single representation level, PhishDetect constructs character-level, word-level, and character–word-level embeddings and maps them to specialized Char-CNN, LSTM, and Bi-LSTM branches, respectively. The Char-CNN captures fine-grained lexical and structural patterns, the LSTM models sequential dependencies among URL tokens, and the Bi-LSTM learns bidirectional contextual relationships from the combined character–word representation. The heterogeneous branch features are subsequently integrated into a unified representation and processed by an attention-based feature weighting mechanism that emphasizes discriminative information before classification. Experiments were conducted using the public Phishing Site URLs dataset, from which a balanced subset of 120,000 URLs was used for model development and evaluation. Feature-dimensionality analysis identified a 155-dimensional fused representation as the most effective configuration. The proposed framework achieved 97.06% accuracy, 96.75% precision, 97.38% recall, and a 97.07% F1-score. Component-level ablation experiments further demonstrated the complementary contribution of the multi-branch architecture and attention mechanism. These findings indicate that representation-aware multi-level learning and adaptive feature weighting provide an effective approach for phishing URL detection.

Keywords: Phishing URL Detection; Deep Learning; Attention Mechanism; Feature Fusion; Multi-Level Embedding

1 Introduction

Phishing has emerged as one of the most prevalent and damaging cyber threats in today's internet ecosystem, posing serious risks to both individuals and organizations [1]. By impersonating legitimate websites, attackers deceive users into disclosing sensitive information such as login credentials, personal data, and financial details [2], [3]. Despite continuous advances in cybersecurity technologies, phishing attacks remain highly effective due to their evolving strategies and low deployment cost.

Among various phishing detection approaches, URL-based detection has attracted increasing attention because it does not require access to webpage content or external resources, enabling rapid and lightweight analysis [3], [4]. Traditional methods, including blacklist/whitelist mechanisms and handcrafted feature-based machine learning models, suffer from inherent limitations [5], [6]. Blacklist-based approaches cannot detect previously unseen (zero-day) phishing URLs, while feature-engineered machine learning methods rely heavily on domain expertise and require frequent updates to remain effective.

Recent advances in deep learning have enabled automatic feature learning directly from raw URLs, significantly improving detection performance [3], [7]. Character-level convolutional neural networks (Char-CNNs) [8] have demonstrated strong capability in capturing local structural patterns, while recurrent neural networks such as Long Short-Term Memory (LSTM) [9] and Bidirectional LSTM (Bi-LSTM) [10] are effective at modeling sequential dependencies. However, models relying on a single representation level may fail to fully capture the complex semantic and structural characteristics of phishing URLs, which often involve subtle character substitutions, misleading word combinations, and contextual dependencies. In addition, attention mechanisms have been widely adopted in deep learning models to enhance feature discrimination by dynamically assigning higher importance to informative features while suppressing less relevant ones [11]. By enabling models to selectively focus on salient components of learned representations, attention mechanisms have demonstrated effectiveness in improving both performance and robustness in sequence classification tasks [12][13].

To address these limitations, this study proposes PhishDetect, a representation-aware multi-branch deep learning framework for phishing URL detection. The main novelty lies in explicitly learning complementary URL characteristics at multiple representation levels rather than relying on a single character- or word-level representation. PhishDetect constructs character-level, word-level, and character–word-level embeddings and assigns them to

specialized Char-CNN, LSTM, and Bi-LSTM branches, respectively. The Char-CNN branch captures fine-grained lexical and structural patterns, the LSTM branch models sequential dependencies among segmented URL tokens, and the Bi-LSTM branch learns bidirectional contextual relationships from the combined character–word representation. This representation-to-architecture mapping enables complementary structural and contextual characteristics of URLs to be learned simultaneously within a unified framework.

A further contribution is the integration of the heterogeneous branch representations into a unified feature space followed by attention-based adaptive feature weighting. Instead of treating all fused features as equally informative, the attention mechanism emphasizes discriminative components before final classification. The framework also systematically investigates branch-level feature dimensionality and identifies a 155-dimensional fused representation (35 Char-CNN + 60 LSTM + 60 Bi-LSTM) as the most effective among the evaluated configurations.

The main contributions of this study are summarized as follows:

- A multi-level URL representation strategy is developed using character-level, word-level, and character–word-level embeddings to capture complementary lexical, structural, and contextual information from URLs.
- A representation-aware parallel architecture is introduced in which Char-CNN, LSTM, and Bi-LSTM are associated with different embedding levels, enabling specialized learning of local character patterns, sequential token dependencies, and bidirectional character–word contextual relationships.
- An attention-based feature fusion mechanism is incorporated after multi-branch integration to adaptively emphasize discriminative components of the unified representation rather than relying solely on conventional feature concatenation.
- The effectiveness of the proposed architecture is empirically validated through feature-dimensionality analysis and component-level ablation experiments. The selected 155-dimensional representation achieves 97.06% accuracy and a 97.07% F1-score, while the ablation results demonstrate the complementary contribution of the individual branches and attention mechanism.

2 Related Work

Phishing detection has been extensively studied in the fields of cybersecurity and machine learning. Existing approaches can generally be categorized into blacklist/heuristic-based

methods, feature-engineered machine learning methods, and deep learning–based methods that learn representations directly from raw URLs or webpage-related signals [14]. This section reviews these approaches and highlights their limitations.

2.1 Blacklist- and Heuristic-Based Detection

Early phishing detection systems primarily relied on blacklist and whitelist mechanisms, which identify malicious URLs by matching them against known phishing databases [6], [15]. Although these approaches are computationally efficient and easy to deploy, they are ineffective against zero-day phishing URLs, which are often short-lived and rapidly generated [16]. Heuristic-based methods attempt to overcome this limitation by defining handcrafted rules, such as abnormal URL length, excessive use of special characters, suspicious subdomain structures, and lexical anomalies [14]. However, such rules are highly dependent on expert knowledge and require frequent updates to keep pace with evolving phishing strategies [17].

2.2 Feature-Engineered Machine Learning Methods

To improve generalization beyond blacklists, many studies formulated phishing detection as a supervised classification problem using manually extracted features [18], [19]. Commonly used features include lexical features (e.g., character distributions and token statistics), host-based features (e.g., WHOIS information and domain age), and content/network features (e.g., HTML structure and redirection behavior) [20-24].

Traditional classifiers such as Support Vector Machines (SVM), Random Forests, Logistic Regression, and Gradient Boosting have achieved promising results when combined with carefully designed features [25], [26]. Nevertheless, these approaches suffer from several drawbacks:

1. performance is highly sensitive to the quality of handcrafted features [27].
2. feature extraction often incurs additional computational and maintenance costs [28].
3. host- and content-based features may require external queries, limiting real-time deployment [29].

2.3 Deep Learning for URL-Based Phishing Detection

Recent advances in deep learning have enabled models to automatically learn discriminative representations directly from raw URLs, reducing reliance on manual feature engineering [30]. Character-level convolutional neural networks (Char-CNNs) treat URLs as character sequences and capture local patterns such as obfuscation cues, special symbols, and subtle character substitutions [31]. These models have demonstrated strong performance in

detecting phishing URLs, particularly those involving character-level manipulation [3].

Recurrent neural networks (RNN), including LSTM and Bi-LSTM, have also been applied to phishing detection to model sequential dependencies within URLs [10]. Bidirectional architectures further improve performance by incorporating both forward and backward contextual information [32]. In addition, some studies transformed URLs into numerical representations (e.g., ASCII-based encoding) and reshaped them into matrix-like inputs for convolutional processing, showing the effectiveness of character-aware representations [33]. Despite these advances, many deep learning approaches rely on a single representation level, such as characters or tokens alone. However, phishing URLs often combine multiple manipulation strategies, including character substitutions, misleading word composition, and structural deception, which cannot be fully captured by a single-level representation [3].

2.4 Hybrid Architectures and Attention Mechanisms

To address the limitations of single-representation models, hybrid architectures that combine CNNs and RNNs have been proposed [2], [34]. These models leverage CNNs for local pattern extraction and RNNs for sequential dependency modeling, achieving improved detection performance [35]. More recently, attention mechanisms have been introduced to enhance feature discrimination by dynamically weighting informative components of learned representations [11], [36]. The additive attention mechanism proposed by Bahdanau et al. [11] has been widely adopted due to its effectiveness and interpretability. In phishing detection tasks, attention mechanisms help highlight salient URL segments or feature dimensions that contribute most to the classification decision [2], [36].

Summary, existing phishing detection methods either rely on blacklists and heuristics with limited zero-day capability [16], [37], depend on handcrafted features with high maintenance cost [38], or utilize deep learning models based on a single representation level [39]. Although hybrid architectures and attention mechanisms have shown promise [11], [40], effectively integrating multi-level URL representations remains an open challenge. These limitations motivate the proposed approach, which combines character-level, word-level, and character–word-level embeddings with an attention-based classifier to capture complementary features and improve robustness.

3 Method

Fig. 1 illustrates the overall workflow of the proposed model. The input URL is first

transformed into embedding representations and processed by convolutional and recurrent neural network modules to extract local structural and sequential features. An attention mechanism is then applied to adaptively weight the extracted features, emphasizing the most informative components. The attended feature representation is finally fed into the classifier to generate the prediction.

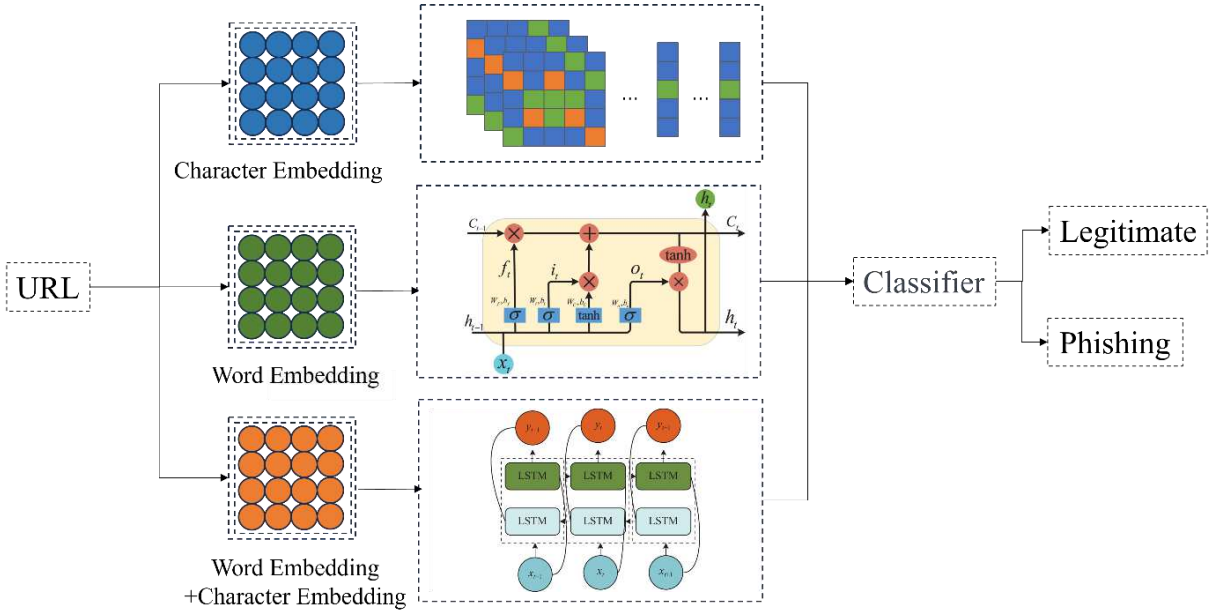


Fig. 1. Overall workflow of the proposed phishing URL detection framework.

3.1 Data Description

The dataset used in this study was obtained from the public Kaggle repository titled Phishing Site URLs developed by Tarun Tiwari. It contains a total of 549,343 URL samples, including 392,923 phishing URLs and 156,420 legitimate URLs. The data are provided in simple text form, where each URL entry is associated with a binary label indicating its category.

3.2 Data Preprocessing

As the dataset consists exclusively of URLs and associated tags, effective URL segmentation and dictionary construction are required prior to model training. Conventional NLP tokenization tools (e.g., NLTK, SpaCy, StanfordNLP) rely on explicit delimiters such as spaces and punctuation and therefore perform poorly on URLs, which are composed of continuous character sequences. To address this limitation, this study adopts the Bidirectional Maximum Matching (BMM) algorithm for URL segmentation, together with the base English dictionary provided by the *pyspellchecker* package. Although this dictionary contains many

common English words, it excludes many URL-specific and technical tokens (e.g., *"www"*, *"https"*), which are essential for accurate segmentation. These tokens are therefore incorporated during preprocessing.

The BMM algorithm, originally developed for Chinese word segmentation, is well suited to URL processing due to the absence of explicit word boundaries. BMM combines Forward Maximum Matching (FMM) and Reverse Maximum Matching (RMM) to improve segmentation robustness. A maximum matching length was determined empirically by analyzing the *pyspellchecker* dictionary (70,317words). The longest entry (*"supercalifragilisticexpialidocious"*) contains 36 characters; accordingly, the maximum word length was set to 36. Prior to segmentation, URLs are lightly preprocessed by splitting on delimiters such as "." and "/". A frequency analysis is then performed on the resulting tokens. Any token appearing more than five times is treated as a valid word and added to the dictionary to ensure that frequently occurring URL components (e.g., *"www"*, *"com"*) are retained.

During segmentation, FMM processes strings from left to right, while RMM operates from right to left, with both methods attempting to match the longest possible substring against the dictionary at each step. Each segmented token is validated against the dictionary and labeled as 1 if it exists in the dictionary and 0 otherwise. The final segmentation output is determined by comparing the results of FMM and RMM: the segmentation containing fewer tokens labeled as 0 is preferred; if both results contain the same number of invalid tokens, the segmentation with fewer total tokens is selected. In the case of a tie, one result is selected arbitrarily. Newly identified tokens that appear more than once are added to the dictionary to iteratively improve segmentation coverage.

3.3 Data Split

The original dataset exhibits a substantial class imbalance. To provide a controlled evaluation and prevent the majority class from dominating model training, a balanced subset was constructed by randomly selecting 60,000 phishing URLs and 60,000 legitimate URLs, resulting in a total of 120,000 samples. The balanced dataset was first divided into training and test sets using an 80:20 stratified split, resulting in 96,000 training samples and 24,000 test samples. Subsequently, 20% of the training data was reserved as a validation set, resulting in 76,800 training samples, 19,200 validation samples, and 24,000 test samples. The training set was used for parameter optimization, the validation set was used for hyperparameter and model-configuration selection, and the independent test set was used only for final performance evaluation.

3.4 Embedding

To capture URL characteristics at different representation levels, three types of embeddings are employed: character embedding, word embedding, and character–word embedding. These embeddings enable the model to learn complementary information from URLs and improve robustness against obfuscated patterns commonly found in phishing URLs.

3.4.1 Character Embedding

3.4.2 Word Embedding

Words are obtained using the BMM segmentation algorithm described in Section 2.2. Unlike characters, the number of words depends on the dataset and can grow significantly. To ensure consistent input size, URLs are padded or truncated to a maximum length of 200 tokens, and each word is embedded into a 32-dimensional vector. To reduce vocabulary size and memory consumption, words that appear only once are replaced with the <UNK> token. In addition, special symbols (e.g., *"%"*, *""* *, *"("*) are treated as independent tokens and explicitly included in the vocabulary, as such symbols often convey important structural information in phishing URLs.

3.4.3 Character–Word Embedding

Character-level and word-level information are jointly exploited by combining character embeddings and word embeddings through element-wise addition. Two embedding matrices are maintained: a character embedding matrix EM_C and a word embedding matrix EM_W, both with a fixed length of 200. After embedding, the character–word representation is obtained as:

$$URL_{CW}=URL_C+URL_W. \quad (1)$$

This fusion strategy is computationally efficient and allows the model to integrate complementary information from different representation levels without increasing embedding dimensionality. Moreover, the character–word embedding improves robustness to rare and unseen words by leveraging character-level cues to supplement word-level representations [24].

3.5 Model Architecture

3.5.1 Char-CNN

Convolutional Neural Networks (CNNs) are a class of deep learning models originally developed for visual pattern analysis and later extended to various sequence modeling tasks. A typical CNN architecture consists of convolutional layers, pooling layers, a flatten layer, and fully connected layers. Through hierarchical feature extraction, CNNs can effectively learn discriminative representations from raw input data. Character-level CNN (Char-CNN) is a natural language processing model that operates on characters rather than words [41]. In this

approach, text is decomposed into character sequences, and convolution operations are applied to capture local character patterns. Char-CNN is particularly effective for handling rare words, misspellings, and newly generated tokens, as it does not rely on a predefined vocabulary [42].

Since Char-CNN processes text at the character level, it can handle previously unseen words by combining known characters. Compared with word-level models, character-level models significantly reduce vocabulary size and memory requirements. Moreover, Char-CNN is language-independent and well suited for capturing fine-grained character-level features in URLs. Therefore, Char-CNN is adopted in this study to extract character-level representations.

3.5.2 LSTM

Long Short-Term Memory (LSTM) is a special type of Recurrent Neural Network (RNN) that has been widely applied to sequence learning tasks such as action recognition and natural language processing. LSTM is particularly effective at learning long-term dependencies by addressing the vanishing gradient problem commonly encountered in standard RNNs.

An LSTM network is composed of a sequence of LSTM units, each containing a memory cell that enables information to be stored, updated, or discarded over time. This memory mechanism is controlled by three gating units: the input gate i_t, the forget gate f_t, and the output gate o_t. These gates regulate the flow of information based on its learned importance. The gate computations are defined as:

$$f_t = \sigma(W_f h_{t-1} + W_f x_t + b_f) \quad (2)$$

$$i_t = \sigma(W_i h_{t-1} + W_i x_t + b_i) \quad (3)$$

$$o_t = \sigma(W_o h_{t-1} + W_o x_t + b_o) \quad (4)$$

where $\sigma(\cdot)$denotes the sigmoid function. The candidate memory state $\hat{c}_t$is computed as:

$$\hat{c}_t = tanh(W_c h_{t-1} + W_c x_t + b_c) \quad (5)$$

The memory cell state $c_t$is updated by:

$$c_t = f_t \odot c_{t-1} + i_o \odot \tilde{c}_t \quad (6)$$

where $\odot$ denotes the Hadamard (elementwise) product. Finally, the hidden state output is given by:

$$h_t = o_t \tanh(c_t) \quad (7)$$

The gating mechanism of LSTM enables selective forgetting, memory updating, and output generation, allowing the model to effectively capture long-range dependencies in sequential data.

3.5.3 Bi-LSTM

Bidirectional LSTM (Bi-LSTM) consists of two independent LSTM layers: one processes

the sequence in the forward direction, while the other processes it in the backward direction. The outputs of both directions are concatenated at each time step to form the final representation. By considering both past and future contextual information simultaneously, Bi-LSTM captures richer contextual dependencies than unidirectional LSTM. This capability is particularly beneficial for modeling word–character-level dependencies in URLs, where contextual cues may depend on both preceding and succeeding tokens. Consequently, Bi-LSTM provides more comprehensive feature representations and improves model accuracy and stability.

3.5.4 Attention Mechanism

The attention mechanism is an important technique in deep learning that was originally introduced to overcome limitations of sequence-to-sequence models in natural language processing. Its core idea is to dynamically assign different importance weights to different parts of the input based on task relevance, allowing the model to focus on the most informative features.

This study adopts the additive attention mechanism, proposed by Bahdanau et al. [11]. The similarity score between a query vector q and a key vector k_i is computed as:

$$e_i = V^T \tanh\left(W_q q + W_k k_i + b\right) \tag{8}$$

where e_i denotes the similarity score, V^T, W_q, and $W_k$are trainable weight matrices, and $b$is a bias term.

The attention weights are obtained by applying the softmax function to the similarity scores:

$$a_i = \frac{\sum \exp(e_i)}{\sum_j \exp\left(e_j\right)} \tag{9}$$

ensuring that the weights sum to one. The final output is computed as a weighted sum of the value vectors v_i:

$$output = \sum_i a_i v_i \tag{10}$$

By adaptively weighting features based on their relevance, the additive attention mechanism generates a comprehensive feature representation that captures complex relationships among input vectors. This enhances classification performance and provides interpretability by indicating which features contribute most to the final decision.

3.5.5 Proposed Method

The proposed framework adopts a multi-branch feature extraction architecture to learn

complementary representations of phishing URLs from different perspectives. As shown in Fig. 2, the model consists of three parallel branches: Char-CNN, LSTM, and Bi-LSTM. Each branch receives a different embedding representation and is designed to capture distinct characteristics of the input URL.

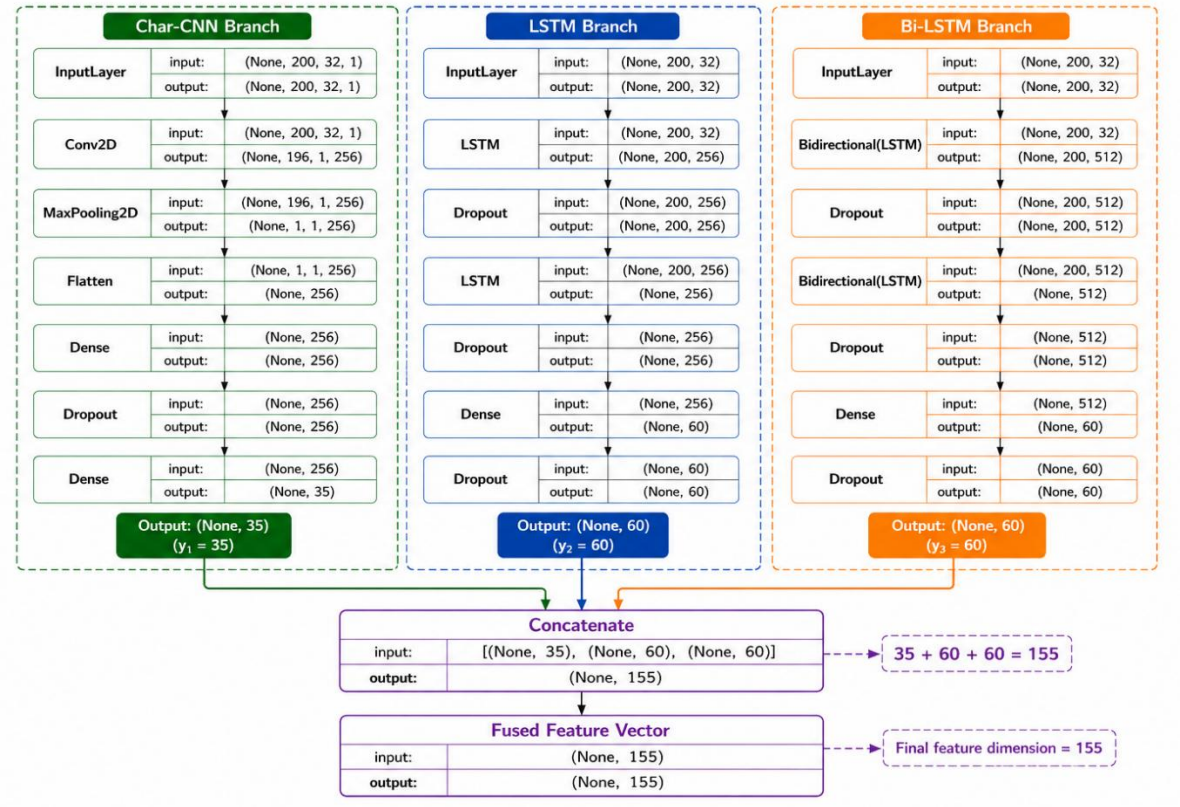


Fig. 2 Proposed Multi-Branch Feature Extraction and Fusion Architecture.

The Char-CNN branch processes the character-level embedding and extracts local structural patterns through a convolutional layer followed by max-pooling, flattening, and fully connected layers. Character-level learning enables the model to identify subtle lexical manipulations commonly used in phishing URLs, such as character substitutions, inserted symbols, and misspellings. The final dense layer produces a 35-dimensional feature vector ($y_1 = 35$). The LSTM branch processes the word-level embedding to model long-range sequential dependencies within the URL. Two stacked LSTM layers are employed with dropout regularization to capture contextual relationships among sequential tokens while reducing overfitting. The extracted representation is transformed into a 60-dimensional feature vector ($y_2 = 60$).

The Bi-LSTM branch operates on the combined character–word embedding and simultaneously learns contextual information in both forward and backward directions. By incorporating bidirectional sequence modeling, the branch captures richer semantic

dependencies and structural relationships than a unidirectional recurrent network. The final dense layer generates a 60-dimensional feature vector ($y_3 = 60$). The outputs of the three branches are concatenated to form a unified 155-dimensional feature vector ($35 + 60 + 60 = 155$). This feature fusion strategy enables the model to integrate complementary local structural patterns, sequential dependencies, and bidirectional contextual information before classification. As demonstrated in the feature dimensionality analysis (Table 1), this configuration achieves the best overall performance among all evaluated settings and is therefore adopted as the final architecture for subsequent experiments.

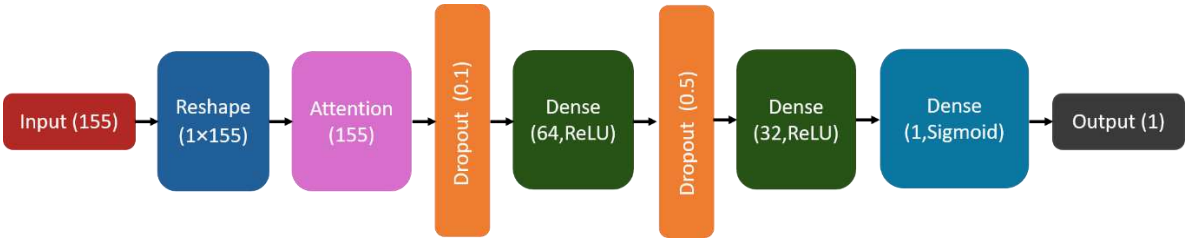


Fig. 3. Proposed Attention-Based Classification Module.

As illustrated in Fig. 3, the fused 155-dimensional feature vector is first reshaped and then fed into an attention layer, which adaptively assigns higher importance weights to the most discriminative features while suppressing less informative ones. This adaptive weighting enables the classifier to focus on the most relevant structural and semantic information extracted from the three feature extraction branches.

The attention-enhanced representation is subsequently passed through a dropout layer and two fully connected layers containing 64 and 32 neurons, respectively, with ReLU activation functions to further refine the learned features and reduce overfitting. Finally, a single-neuron output layer with a sigmoid activation function produces the probability that the input URL belongs to the phishing class. By integrating character-level structural information, sequential contextual dependencies, bidirectional contextual representations, and attention-based feature weighting, the proposed framework effectively captures both local and global characteristics of phishing URLs, resulting in improved detection accuracy and robust generalization.

3.6 Evaluation Metrics

In this study, phishing URL detection is formulated as a binary classification task. Model performance is evaluated using four widely adopted metrics: Accuracy (A), Precision (P), Recall (R), and F1-score (F1). Phishing URLs are treated as the positive class. Accordingly, True Positive (TP) denotes the number of phishing URLs that are correctly classified as phishing, while True Negative (TN) represents the number of legitimate URLs that are

correctly classified as legitimate. False Positive (FP) refers to legitimate URLs that are incorrectly classified as phishing, whereas False Negative (FN) denotes phishing URLs that are incorrectly classified as legitimate.

Accuracy (A) measures the overall classification performance and is defined as the proportion of correctly classified URLs, as shown in Eq. (11). Precision (P) evaluates the correctness of phishing detection by measuring the proportion of correctly identified phishing URLs among all URLs predicted as phishing (Eq. (12)). Recall (R) measures the coverage of actual phishing URLs, indicating the proportion of phishing URLs that are correctly identified (Eq. (13)). The F1-score (F1) provides a balanced measure of precision and recall and is defined in Eq. (14).

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

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

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

$$F1 = 2 * \left(\frac{P \times R}{P+R}\right) \tag{14}$$

4 Experimental Results

This section presents the experimental evaluation of the proposed model. It includes an analysis of hyperparameter settings, an investigation of the impact of the output feature dimensions of the Char-CNN, LSTM, and Bi-LSTM branches, an ablation study of different model combinations, and a comparison with representative baseline architectures to validate the effectiveness of the proposed framework.

4.1 Experimental Setup

All experiments were implemented using TensorFlow 2.12 and conducted on a workstation equipped with an NVIDIA RTX 4090 GPU, an Intel Core i7 processor, and 64 GB of RAM. Phishing URL detection was formulated as a binary classification task. The models were trained using the Adam optimizer with binary cross-entropy as the loss function. The effects of key hyperparameters, including the learning rate, number of training epochs, batch size, and branch-level feature dimensions, were systematically investigated to determine an appropriate configuration for the proposed framework. The selected hyperparameter configuration was subsequently used for the ablation study and comparison with the baseline models.

4.2 Hyperparameter Analysis

This study investigates the impact of key hyperparameters, including the learning rate, number of epochs, and batch size, on model performance. When evaluating a single hyperparameter, all other parameters were fixed at their empirically optimal values. The Adam optimizer was employed for training, and the binary cross-entropy loss function was used.

The learning rate determines the magnitude of parameter updates during model training and can substantially affect convergence behavior and classification performance. Different learning-rate settings were experimentally evaluated to identify an appropriate value for the proposed framework. As shown in Fig. 4, a learning rate of 0.0001 provided the best overall performance among the evaluated settings and was therefore selected for subsequent experiments. The corresponding training-loss curves in Fig. 5 further illustrate the effect of learning-rate selection on model convergence. A comparatively high learning rate resulted in less stable optimization, whereas an excessively low learning rate led to slower convergence. Based on these observations, 0.0001 was adopted as the learning rate for the final model.

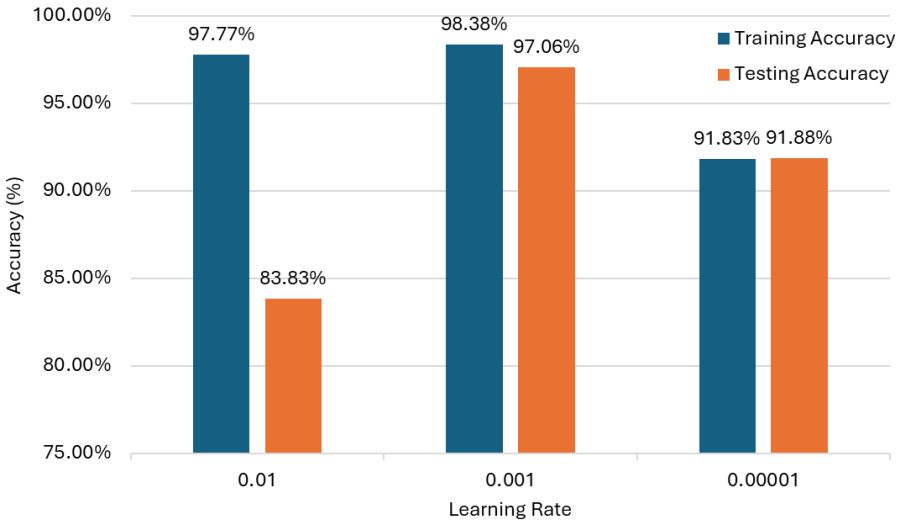
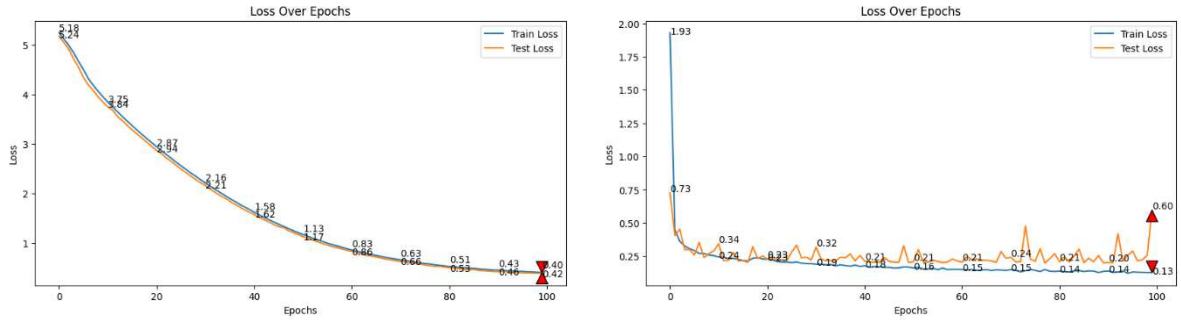


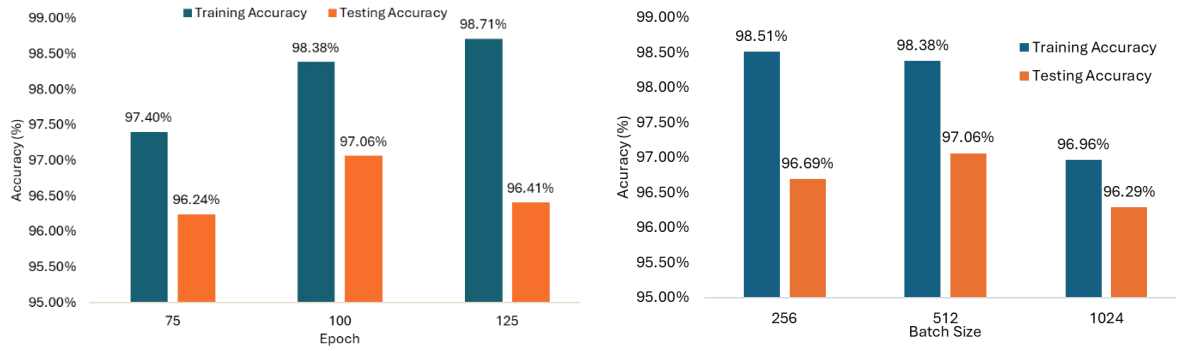
Fig. 4. Comparison of model performance under different learning rates.



(a) **(b)**

Fig. 5. Training loss curves under different learning rates: **(a)** learning rate = 0.00001; **(b)** learning rate = 0.001.

The number of epochs determines how many times the model iterates over the training dataset. As shown in Fig. 6(a), increasing the number of epochs beyond 100 did not improve generalization performance and led to longer training time. Therefore, the number of epochs was set to 100. Batch size affects both training efficiency and generalization. Experimental results in Fig. 6(b) indicate that a batch size of 512 provides a better balance between performance and training efficiency compared with smaller batch sizes. Based on these experiments, the final hyperparameter configuration uses the Adam optimizer, a learning rate of 0.0001, binary cross-entropy loss, 100 epochs, and a batch size of 512.



(a) **(b)**

Fig. 6. Effect of training epochs and batch size on model performance: **(a)** training and testing accuracy for different numbers of epochs; **(b)** training and testing accuracy for different batch sizes.

To determine the optimal output feature dimensions for the three feature extraction branches, experiments were conducted using different combinations of the Char-CNN (y_1), LSTM (y_2), and Bi-LSTM (y_3) output dimensions. The corresponding experimental results are presented in Table 1.

As shown in Table 1, the feature dimension combination of 35 (Char-CNN), 60 (LSTM), and 60 (Bi-LSTM), resulting in a total feature dimension of 155, achieves the best overall performance, with an accuracy of 97.06%, a recall of 97.38%, a precision of 96.75%, and an F1-score of 97.07%. Lower-dimensional configurations (e.g., 140) provide insufficient representational capacity, whereas higher-dimensional configurations (160–205) introduce redundant features without improving classification performance. Increasing the feature dimensions beyond 155 leads to slight reductions in accuracy and F1-score, suggesting that excessive feature dimensions may increase model complexity and reduce generalization capability. Therefore, the 155-dimensional fused feature vector is selected as the optimal

configuration and is adopted in all subsequent experiments.

Table 1. Effect of feature vector dimensionality

Total	Bi-LSTM (y_3)	LSTM (y_2)	Char-CNN (y_1)	A (%)	R (%)	P (%)	F1 (%)
140	55	55	30	96.84	95.85	97.74	96.78
155	**60**	**60**	**35**	**97.06**	**97.38**	**96.75**	**97.07**
160	65	65	40	96.77	98.06	95.65	96.84
175	70	70	45	96.82	98.05	95.70	96.86
190	75	75	50	96.29	94.80	97.74	96.25
205	85	85	60	95.78	98.97	94.08	95.94

4.3 Ablation Study of Model Combinations

An ablation study was conducted to evaluate the contribution of each component in the proposed framework by comparing different architectural combinations. The experimental results are summarized in Table 2.

As shown in Table 2, the single-component models (Char-CNN, LSTM, and Bi-LSTM) achieve competitive performance, with Bi-LSTM producing the best results among the three individual branches (95.10% accuracy and 95.14% F1-score). This indicates that bidirectional sequence modeling captures richer contextual information than either character-level convolution or unidirectional sequence modeling alone. Nevertheless, relying on a single feature extraction strategy is insufficient to fully capture the complex structural and semantic characteristics of phishing URLs. Performance improves consistently when multiple feature extraction branches are combined. Architectures integrating Char-CNN with LSTM or Bi-LSTM outperform the corresponding single-branch models, demonstrating that local character-level patterns and sequential contextual information provide complementary representations. Furthermore, incorporating the attention mechanism consistently improves classification performance by adaptively assigning higher importance to discriminative features while suppressing less informative ones.

The complete model, which integrates Char-CNN, LSTM, Bi-LSTM, and the attention mechanism, achieves the best overall performance, with an accuracy of 97.06%, a precision of 96.75%, a recall of 97.38%, and an F1-score of 97.07%. Compared with the strongest single-component model (Bi-LSTM), the proposed framework improves the F1-score by 1.93 percentage points, confirming that each component contributes meaningfully to the overall architecture. These results demonstrate that combining multi-level feature extraction with attention-based feature fusion effectively leverages complementary information and enhances phishing URL detection performance.

Table 2. Ablation analysis of multi-level URL representations

Model Combination	A (%)	P (%)	R (%)	F1 (%)
Char-CNN	93.31	98.46	87.98	92.93
LSTM	93.76	93.75	93.72	93.78
Bi-LSTM	95.10	95.07	95.20	95.14
Bi-LSTM + LSTM + Attention	94.30	94.85	93.80	94.32
LSTM + Char-CNN + Attention	96.01	97.08	94.97	96.02
Bi-LSTM + Char-CNN + Attention	96.50	97.16	95.84	96.49
Bi-LSTM + LSTM + Char-CNN	95.53	94.40	96.83	95.60
Bi-LSTM + LSTM + Char-CNN + Attention	**97.06**	**96.75**	**97.38**	**97.07**

4.4 Comparison with Baseline Architectures

To evaluate the effectiveness of the proposed framework, comparative experiments were conducted against three representative baseline architectures. The performance comparison on the test set is presented in Table 3. The proposed multi-level embedding framework achieves the best overall performance across all evaluation metrics, demonstrating its superior feature representation capability and generalization performance.

The baseline models include a standard CNN-based URL classifier [43], a hybrid CNN-LSTM model [44] , and a Char-CNN model [40]. To ensure a fair comparison, all baseline models were re-implemented according to their original configurations. Minor modifications were applied only when necessary to accommodate the characteristics of the dataset used in this study. Specifically, the maximum URL length of the CNN-LSTM model was increased from 75 characters, as reported in the original work, to 200 characters to maintain consistency with the preprocessing strategy adopted in this study.

As shown in Table 3, the proposed model achieves the highest performance, with an accuracy of 97.06%, precision of 96.75%, recall of 97.38%, and an F1-score of 97.07%. Compared with the strongest baseline model (CNN-LSTM), the proposed framework improves the accuracy by 0.76 percentage points and the F1-score by 0.73 percentage points. Compared with the Char-CNN baseline, the proposed model improves the F1-score by 1.56 percentage points, while substantially outperforming the conventional CNN model across all evaluation metrics. These improvements demonstrate that integrating multi-level embedding, parallel feature extraction, and attention-based feature fusion provides more discriminative feature representations than conventional single-branch or hybrid architectures, resulting in more accurate and robust phishing URL detection.

Table 3. Performance comparison with other models

Model	A (%)	P (%)	R (%)	F1 (%)
CNN	92.36	92.20	90.22	92.20
Char-CNN	95.45	94.26	96.80	95.51
CNN-LSTM	96.30	94.50	98.25	96.34
Proposed Model	**97.06**	**96.75**	**97.38**	**97.07**

5 Discussion

The experimental results demonstrate that the proposed multi-level embedding framework consistently outperforms the baseline and hybrid models across all evaluation metrics. This improvement is primarily attributed to the integration of character-level, word-level, and character–word-level representations, which enables the framework to capture complementary structural and semantic information from phishing URLs. Since phishing URLs often contain both subtle lexical manipulations and complex contextual relationships, combining multiple representation levels allows the model to learn more discriminative features than a single-branch architecture.

The feature dimensionality analysis further shows that a total feature dimension of 155 provides the best balance between representational capacity and model complexity. Smaller feature dimensions cannot adequately capture the discriminative characteristics of phishing URLs, whereas larger feature dimensions introduce redundant information that reduces generalization performance. These findings indicate that selecting an appropriate feature dimension is essential for achieving high detection accuracy while maintaining an efficient feature representation.

The ablation study demonstrates that each feature extraction branch contributes uniquely to the overall performance of the proposed framework. Among the individual branches, the Bi-LSTM provides the strongest sequential representation by utilizing contextual information from both forward and backward directions. However, the results also show that relying on a single feature extraction strategy is insufficient because phishing URLs simultaneously exhibit local character manipulations and broader structural relationships. Integrating the Char-CNN, LSTM, and Bi-LSTM branches enables the framework to exploit complementary information and produce a richer feature representation.

When convolutional and recurrent components are combined, performance improves consistently. Architectures integrating Char-CNN with LSTM or Bi-LSTM outperform the

corresponding single-branch models, demonstrating that hybrid feature extraction provides more discriminative feature representations. This finding is consistent with previous CNN–RNN-based phishing and malicious URL detection approaches, which reported improved robustness by jointly exploiting local and sequential features [15], [45], [46], [47].

The attention-based feature fusion mechanism further enhances the effectiveness of the proposed framework. By dynamically assigning greater importance to informative features while suppressing less relevant ones, the attention mechanism produces a more discriminative fused representation for classification. This adaptive feature weighting enables the classifier to focus on the most relevant structural and semantic characteristics extracted from the three feature extraction branches, thereby improving both detection performance and model robustness.

Comparative experiments with the baseline architectures demonstrate that the proposed framework achieves higher accuracy and F1-score than the CNN, Char-CNN, and CNN–LSTM models. These results confirm that combining multi-level embedding with attention-based feature fusion provides a more comprehensive representation of phishing URLs than conventional single- or dual-branch architectures. Unlike traditional feature-engineered approaches that rely on manually designed features and external resources, the proposed framework automatically learns discriminative representations directly from raw URLs, making it more adaptable to emerging phishing strategies. These characteristics make the proposed framework well suited for practical phishing URL detection systems, where the ability to automatically learn robust representations from evolving URL patterns is essential for maintaining high detection performance.

6 Limitations and Future Work

Although the proposed multi-level embedding framework demonstrates strong performance in phishing URL detection, several limitations should be acknowledged. First, the experimental evaluation is conducted on a single public dataset. While the dataset is balanced and widely used, it may not fully capture the diversity and evolving nature of real-world phishing URLs. Phishing strategies vary across regions, languages, and time periods, and model performance may differ when applied to unseen datasets with different distributions. Furthermore, the dataset was balanced to provide a controlled evaluation of the proposed framework. Although this setting facilitates fair model comparison, it does not fully reflect the severe class imbalance encountered in practical phishing detection. Future studies will evaluate the proposed framework on naturally imbalanced datasets to better assess its robustness and

real-world applicability.

Second, the proposed architecture integrates multiple deep learning components, including Char-CNN, LSTM, Bi-LSTM, and an attention mechanism. Although this design improves detection performance, it also increases computational complexity and memory consumption. This may limit deployment in resource-constrained environments, such as edge devices or real-time large-scale filtering systems.

Third, the current study focuses exclusively on URL-based features and does not incorporate auxiliary information such as webpage content, host-based attributes, or network-level signals. While URL-only detection offers efficiency and ease of deployment, combining multiple data sources may further enhance robustness against sophisticated phishing attacks.

Future research will address these limitations in several directions. First, cross-dataset and cross-domain evaluations will be conducted to assess the generalization capability of the proposed model under diverse and evolving phishing scenarios. Second, lightweight and efficient variants of the proposed architecture will be explored to reduce computational overhead while maintaining detection accuracy. Third, future work may investigate multimodal phishing detection frameworks that integrate URL-based features with content-level and host-based information. Additionally, online learning and continual learning strategies will be considered to enable the model to adapt dynamically to newly emerging phishing patterns.

7 Conclusion

This study presented PhishDetect, a representation-aware multi-branch deep learning framework for phishing URL detection. The framework was designed to address the limitations of single-level URL representations by jointly exploiting character-level, word-level, and character–word-level information. Rather than applying multiple learning architectures to an identical representation, the proposed approach maps each representation level to a specialized feature-learning branch: Char-CNN captures fine-grained lexical and structural patterns, LSTM models sequential dependencies among URL tokens, and Bi-LSTM learns bidirectional contextual relationships from the combined character–word representation. The resulting heterogeneous features are integrated into a unified representation and subsequently processed using an attention mechanism to adaptively emphasize discriminative information before classification.

Experimental analysis identified a 155-dimensional fused feature representation, comprising 35 Char-CNN, 60 LSTM, and 60 Bi-LSTM features, as the most effective among the evaluated configurations. The complete framework achieved 97.06% accuracy, 96.75%

precision, 97.38% recall, and a 97.07% F1-score. The component-level ablation study further demonstrated that the complete multi-branch architecture with attention outperformed the individual branches and alternative branch combinations, supporting the complementary role of multi-level representations and adaptive feature weighting. These findings demonstrate the effectiveness of explicitly coupling heterogeneous URL representations with specialized feature extractors for phishing URL classification.

Despite these promising results, the current evaluation is based on a single public dataset and a balanced experimental subset; therefore, broader generalization to naturally imbalanced, temporally evolving, and cross-domain phishing data remains to be established. Future work will investigate cross-dataset and temporal validation, lightweight architectures for real-time deployment, adversarial robustness, and continual learning to improve adaptability to emerging phishing strategies.

Declarations

Clinical Trial Registration

Not applicable. This study focuses on phishing URL detection using deep learning techniques and does not involve any clinical trials.\

Funding. This research received no external funding.

Data Availability. The dataset used in this study is publicly available from Kaggle under the title *Phishing Site URLs*, developed by Tarun Tiwari. The dataset can be accessed at: [INSERT THE VERIFIED KAGGLE DATASET URL]. https://www.kaggle.com/datasets/taruntiwarihp/phishing-site-urls

Ethics, Consent to Participate, and Consent to Publish Declarations

Not applicable. This study involves the development and evaluation of a deep learning framework for phishing URL detection and does not involve human participants, human subject experiments, or identifiable personal data.

References

[1] C.Lai, A.BinSelamat, R. B.Ibrahim, N.Do, H.Fujita, andO.Krejcar, “URL Phishing Detection by Using Natural Language Processing and Deep Learning Model,” *Front. Artif. Intell. Appl.*, vol. 389, pp. 110–119, 2024, doi: 10.3233/FAIA240360.

[2] N.Do, A.BinSelamat, O.Krejcar, andH.Fujita, “Detection of malicious URLs using Temporal Convolutional Network and Multi-Head Self-Attention mechanism,” *Appl. Soft Comput.*, vol. 169, 2025, doi: 10.1016/j.asoc.2024.112540.

[3] K.Sruthi andS.Manohar Naik, “A novel framework for phishing attack detection using

domain-adapted GloVe embeddings and attention-enhanced neural sequence model," *Appl. Soft Comput.*, vol. 188, 2026, doi: 10.1016/j.asoc.2025.114441.

[4] V.Ravula andM.Ramaiah, "Enhancing phishing detection with dynamic optimization and character-level deep learning in cloud environments," *PeerJ Comput. Sci.*, vol. 11, 2025, doi: 10.7717/peerj-cs.2640.

[5] R. S.Rao, C.Kondaiah, A. R.Pais, andB.Lee, "A hybrid super learner ensemble for phishing detection on mobile devices," *Sci. Rep.*, vol. 15, no. 1, 2025, doi: 10.1038/s41598-025-02009-8.

[6] N. A.Azeez, S.Misra, I. A.Margaret, L.Fernandez-Sanz, andS. M.Abdulhamid, "Adopting automated whitelist approach for detecting phishing attacks," *Comput. Secur.*, vol. 108, p. 102328, Sep.2021, doi: 10.1016/j.cose.2021.102328.

[7] S. K.Birthriya, P.Ahlawat, andA. K.Jain, "Enhanced Phishing Website Detection Using Dual-Layer CNN and GRU with Attention Mechanism and Lexical NLP Features," *SN Comput. Sci.*, vol. 5, no. 7, 2024, doi: 10.1007/s42979-024-03282-6.

[8] M.Almohaimeed, F.Albalwy, L.Algulaiti, andH.Althubyani, "Phishing URL Detection Using Deep Learning: A Resilient Approach to Mitigating Emerging Cybersecurity Threats," *Ing. des Syst. d'Information*, vol. 30, no. 5, pp. 1219–1227, 2025, doi: 10.18280/isi.300510.

[9] J.Sotomayor, N.García, andW.Ticona, "Proposal of a Model Based on Deep Learning Techniques for the Detection of Phishing in Social Networks," *Lect. Notes Networks Syst.*, vol. 1559 LNNS, pp. 225–244, 2025, doi: 10.1007/978-3-032-00712-4_15.

[10] S. S.Roy, A. I.Awad, L. A.Amare, M. T.Erkihun, andM.Anas, "Multimodel Phishing URL Detection Using LSTM, Bidirectional LSTM, and GRU Models," *Futur. Internet*, vol. 14, no. 11, p. 340, Nov.2022, doi: 10.3390/fi14110340.

[11] D.Bahdanau, K. H.Cho, andY.Bengio, "Neural machine translation by jointly learning to align and translate," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, 2015.

[12] I.Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, Jan.2017, [Online]. Available: https://ieeexplore.ieee.org/document/974352/

[13] Z.Lin *et al.*, “A structured self-attentive sentence embedding,” *5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc.*, pp. 1–15, 2017.

[14] S.Kailas andR.Roopalakshmi, “‘Think Before You Click’—Malicious URL Detection in Cybersecurity: A Systematic Review and Research Roadmap,” *IEEE Access*, vol. 13, pp. 154305–154325, 2025, doi: 10.1109/ACCESS.2025.3601387.

[15] L.Meng, Z.Ma, andE.Zhu, “Phishing Detection Model Integrating URL Characters and HTML Word Semantic Deep Features,” Institute of Electrical and Electronics Engineers Inc., 2024, pp. 468–473. doi: 10.1109/ICCTIT64404.2024.10928669.

[16] E.Kytidou, T.Tsikriki, G.Drosatos, andK.Rantos, “Machine learning techniques for phishing detection: A review of methods, challenges, and future directions,” *Intell. Decis. Technol.*, vol. 19, no. 6, pp. 4356–4379, Nov.2025, doi: 10.1177/18724981251366763.

[17] L.Zhou, A.Gaurav, W. S.Alhalabi, V.Arya, andE. A.Alharbi, “Integrating AI and Semantic Web Technologies for Robust Phishing Detection in Virtual Realities,” *Int. J. Semant. Web Inf. Syst.*, vol. 21, no. 1, 2025, doi: 10.4018/IJSWIS.371415.

[18] G. S.Nayak, B.Muniyal, andM. C.Belavagi, “Enhancing Phishing Detection: A Machine Learning Approach With Feature Selection and Deep Learning Models,” *IEEE Access*, vol. 13, pp. 33308–33320, 2025, doi: 10.1109/ACCESS.2025.3543738.

[19] S.Salloum, T.Gaber, S.Vadera, andK.Shaalan, “A Systematic Literature Review on Phishing Email Detection Using Natural Language Processing Techniques,” *IEEE Access*, vol. 10, pp. 65703–65727, 2022, doi: 10.1109/ACCESS.2022.3183083.

[20] T.Patel andD.Rathod, “Enhanced Malicious URL Detection Through Feature Significance Analysis,” in *Advances in Smart Computing and Applications*, P.Buyya, R., Gandhi, S., Chaubey, N., Arolkar, H., Agrawal, Ed., Springer, Cham, 2026, ch. Communicat, pp. 108–133. doi: 10.1007/978-3-032-00350-8_9.

[21] G.Chen, W.Le, Z.Jia, J.Li, G.Zhou, andJ.Wang, “TSRA-Net: A pest detection method based on token statistics and reparameterized adaptive networks,” *Expert Syst. Appl.*, vol. 304, p. 130659, Apr.2026, doi: 10.1016/j.eswa.2025.130659.

[22] L.Khedekar andS.Pawar, “Enhanced Feature Extraction for Phishing URL Detection: A Comprehensive Analysis of Structural, Host-Based, Content and N-Gram

Attributes," in *ICT Analysis and Applications*, A.Fong, S., Dey, N., Joshi, Ed., Springer, Cham, 2026, pp. 465–475. doi: 10.1007/978-3-032-06700-5_47.

[23] Y.Cheng, T.Chai, Z.Zhang, K.Lu, andY.Du, "Detecting Malicious Domain Names with Abnormal WHOIS Records Using Feature-Based Rules," *Comput. J.*, vol. 65, no. 9, pp. 2262–2275, Sep.2022, doi: 10.1093/comjnl/bxab062.

[24] K. P. R. D. A.Mathew, "Phishing Detection: Exploring Feature Representation and Defense Strategies," in *Proceedings of the 4th International Conference on Signal and Data Processing*, 2026. doi: 10.1007/978-981-95-1058-0.

[25] Y.Ari Kustiawan andK. I.Ghauth, "Evaluating the Impact of Feature Engineering in Phishing URL Detection: A Comparative Study of URL, HTML, and Derived Features," *IEEE Access*, vol. 13, pp. 126756–126768, 2025, doi: 10.1109/ACCESS.2025.3579223.

[26] J. D.Duarte *et al.*, "Machine Learning for Early Detection of Phishing URLs in Parked Domains: An Approach Applied to a Financial Institution," *IEEE Access*, vol. 13, pp. 145736–145753, 2025, doi: 10.1109/ACCESS.2025.3599454.

[27] H.Al-Tairi, A.Javed, T.Khan, andA. K. J.Saudagar, "DeepLASD countermeasure for logical access audio spoofing," *Sci. Rep.*, vol. 15, no. 1, p. 20839, Jul.2025, doi: 10.1038/s41598-025-04808-5.

[28] S.Mardieva, S.Ahmad, S.Umirzakova, M. J. A.Rasool, andT. K.Whangbo, "Lightweight image super-resolution for IoT devices using deep residual feature distillation network," *Knowledge-Based Syst.*, vol. 285, p. 111343, Feb.2024, doi: 10.1016/j.knosys.2023.111343.

[29] Y.Gong andK.Cai, "A Containerized Microservices Architecture with Reinforcement Learning for Scalable, Adaptive Learning," *J. Web Eng.*, Dec.2025, doi: 10.13052/jwe1540-9589.2482.

[30] N. Q.Do, A.Selamat, O.Krejcar, E.Herrera-Viedma, andH.Fujita, "Deep Learning for Phishing Detection: Taxonomy, Current Challenges and Future Directions," *IEEE Access*, vol. 10, pp. 36429–36463, 2022, doi: 10.1109/ACCESS.2022.3151903.

[31] J.Wu, Z.Yang, L.Guo, Y.Li, andW.Liu, "Convolutional Neural Network with Character Embeddings for Malicious Web Request Detection," in *2019 IEEE Intl Conf*

on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), IEEE, Dec. 2019, pp. 622–627. doi: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00094.

[32] N.Do, A.BinSelamat, K. C.Lim, O.Krejcar, andN. A.Md Ghani, "Transformer-Based Model for Malicious URL Classification," Institute of Electrical and Electronics Engineers Inc., 2023, pp. 323–327. doi: 10.1109/ICOCO59262.2023.10397705.

[33] J.Yuan, G.Chen, S.Tian, andX.Pei, "Malicious URL Detection Based on a Parallel Neural Joint Model," *IEEE Access*, vol. 9, pp. 9464–9472, 2021, doi: 10.1109/ACCESS.2021.3049625.

[34] F. S.Alsubaei, A. A.Almazroi, andN.Ayub, "Enhancing Phishing Detection: A Novel Hybrid Deep Learning Framework for Cybercrime Forensics," *IEEE Access*, vol. 12, pp. 8373–8389, 2024, doi: 10.1109/ACCESS.2024.3351946.

[35] E.Kritika, "A comprehensive literature review on phishing URL detection using deep learning techniques," *J. Cyber Secur. Technol.*, vol. 9, no. 4, pp. 315–343, Oct.2025, doi: 10.1080/23742917.2024.2378552.

[36] J.K S andA.B, "Exploring GRU-based approaches with attention mechanisms for accurate phishing URL detection," *Intell. Decis. Technol.*, vol. 18, no. 2, pp. 1029–1052, Jun.2024, doi: 10.3233/IDT-240026.

[37] M.Abdolrazzagh-Nezhad andN.Langarib, "Phishing Detection Techniques: A review," *Data Sci. J. Comput. Appl. Informatics*, vol. 9, no. 1, pp. 32–46, Jan.2025, doi: 10.32734/jocai.v9.i1-19904.

[38] A.Kurniawan, T. R.Soeprobowati, andB.Warsito, "Hybrid PSO-ACO Optimization for Rice Leaf Disease Classification Using Random Forest and Support Vector Machines," *Int. J. Adv. Comput. Sci. Appl.*, vol. 16, no. 6, pp. 388–397, 2025, doi: 10.14569/IJACSA.2025.0160638.

[39] F.Zheng, Q.Yan, V. C. M.Leung, F. R.Yu, andZ.Ming, "HDP-CNN: Highway deep pyramid convolution neural network combining word-level and character-level representations for phishing website detection," *Comput. Secur.*, vol. 114, 2022, doi: 10.1016/j.cose.2021.102584.

[40] R.Yang, K.Zheng, B.Wu, C.Wu, andX.Wang, “Phishing Website Detection Based on Deep Convolutional Neural Network and Random Forest Ensemble Learning,” *Sensors*, vol. 21, no. 24, p. 8281, Dec.2021, doi: 10.3390/s21248281.

[41] X.Zhang, J.Zhao, andY.Lecun, “Character-level convolutional networks for text classification,” *Adv. Neural Inf. Process. Syst.*, vol. 2015-Janua, pp. 649–657, 2015.

[42] M.Mohamed andS.Al-Azani, “Enhancing Arabic NLP Tasks through Character-Level Models and Data Augmentation,” *Proc. - Int. Conf. Comput. Linguist. COLING*, pp. 2744–2757, 2025.

[43] S. Y.Yerima andM. K.Alzaylaee, “High Accuracy Phishing Detection Based on Convolutional Neural Networks,” *ICCAIS 2020 - 3rd Int. Conf. Comput. Appl. Inf. Secur.*, pp. 0–5, 2020, doi: 10.1109/ICCAIS48893.2020.9096869.

[44] Y.V.M, B.Janet, andS.Reddy, “Anti-phishing System using LSTM and CNN,” in *2020 IEEE International Conference for Innovation in Technology (INOCON)*, IEEE, Nov. 2020, pp. 1–5. doi: 10.1109/INOCON50539.2020.9298298.

[45] B. C.Ujah-Ogbuagu, O. N.Akande, andE.Ogbuju, “A hybrid deep learning technique for spoofing website URL detection in real-time applications,” *J. Electr. Syst. Inf. Technol.*, vol. 11, no. 1, p. 7, Jan.2024, doi: 10.1186/s43067-023-00128-8.

[46] N.Sanjana, N.Kavyasree, andK.Ashwini, “Malicious URL Detection Using Character-Level Tokenization with CNN-LSTM Deep Learning Model,” in *2025 Third International Conference on Networks, Multimedia and Information Technology (NMITCON)*, IEEE, Aug. 2025, pp. 1–6. doi: 10.1109/NMITCON65824.2025.11187823.

[47] W.Wang, F.Zhang, X.Luo, andS.Zhang, “PDRCNN: Precise Phishing Detection with Recurrent Convolutional Neural Networks,” *Secur. Commun. Networks*, vol. 2019, pp. 1–15, Oct.2019, doi: 10.1155/2019/2595794.