INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGIES
BULGARIAN ACADEMY OF SCIENCES
CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **26**, No 1
Sofia • 2026 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2026-0008

PhishFusionNet: A Wide and Deep Phishing Detection with a Hybrid Learning Approach

Ali A. Alani, Adil Al-Azzawia

University of Diyala, College of Science, Computer Science Department, Diyala, Iraq
E-mails: alialani@uodiyala.edu.iq adil_alazzawi@uodiyala.edu.iq

***Abstract*:** *Phishing is a type of cyber threat that targets organizations and individuals worldwide, causing billions of dollars in losses. So far, most successful anti-phishing methods require experts to extract features from phishing sites and third-party detection systems to detect them. This paper presents a PhishFusionNet model, an effective wide-and-deep learning framework for identifying phishing URLs with a high degree of generalization and accuracy. The proposed model successfully discovers both sequential and global patterns within URLs. This is achieved by integrating character-level embeddings that represent the deep component with handcrafted URL features that capture the wide component. We have tested our proposed model on over six million real-world labelled URLs. The results of the test on such a large-scale dataset with an optimal accuracy of 98.9 % have demonstrated that our model outperforms many other tested approaches. Based on these results, we believe that our proposed model is an effective, reliable, and scalable solution for cybersecurity and real-time phishing-detection applications.*

***Keywords*:** *Deep learning, Phishing, Cyber security, Machine learning.*

1. Introduction

Nowadays, the internet is used by most government institutions and private companies to carry out many of their important activities, including financial transactions. The interment helps these institutions increase their accessibility to users and customers. However, many security risks are also introduced by using the internet due to its uncontrollable and anonymous nature. These security risks could enable attackers to compromise these systems more easily and result in severe damage to these organizations and institutions [1]. Phishing attacks remain among the most well-known cyberattacks, in which attackers use social engineering tactics to deceive users [2]. Phishers employ various techniques to deceive users and obtain access to sensitive data such as financial information or user identity.

Most of these attacks range from financial fraud in banking to intellectual property theft in technology and target a wide many sectors, including Technology, Government, Finance, Education, Healthcare, manufacturing, and Retail. Therefore,

there is a pressing need for robust cybersecurity measures to mitigate these risks [1, 2].

According to the Anti-Phishing Working Group (APWG) in the first quarter of 2025, APWG registered 1,003,924 phishing attacks, and this was the highest number since late 2023 [3]. Researchers have developed numerous anti-phishing techniques to detect phishing attacks. B a s i t et al. [4] conducted a comprehensive survey of phishing attacks and classified them by type, channel, and technical characteristics. Additionally, they highlight the need for a reactive detection technique to detect and stop phishing attacks. One of these effective techniques is an Artificial Intelligence (AI) method, such as Deep Learning (DL) and Machine Learning (ML), which are rapidly developing and being used to secure computer operations and manage cybersecurity. For example, ML methods provide good results of accuracy for many benchmark phishing datasets [5-8]; however, these methods have several remarkable restrictions, such as their inability to identify the contextual and semantic patterns within URLs due to their analysis being restricted to predefined features [9].

Recently, Deep learning has emerged as a powerful tool for automatically extracting features from unprocessed data. It is particularly useful for training big data systems or systems without well-defined features. In particular, the development of sequence-modelling capabilities has been greatly aided by Recurrent Neural Networks (RNNs) [10, 11] and Convolutional Neural Networks (CNNs) [12]. In addition, other DL techniques are showing promise for identifying features in observed datasets, such as Bayesian Additive Regression Trees (BART) and Graph Convolutional Networks (GCN) [13]. However, despite DL promising results, DL has a high computational cost. Furthermore, when new websites (or data) become available, the model must be retrained.

1.1. Motivation

Phishing is one of the most pervasive and dangerous cybersecurity threats, deceiving users into divulging their personal information and posing serious risks to their privacy and finances. Many detection techniques currently in use rely on manually engineered features and shallow machine learning models, which often yield high false-positive and false-negative rates and are unable to keep pace with the dynamic structure of contemporary URLs. In addition, the models used in earlier research have often been trained on comparatively small datasets, which restricts their ability to generalize to extensive, real-world scenarios. These limitations highlight the need for a scalable and reliable phishing-detection system.

1.2. Contribution

By optimizing and merging current techniques, this research paper presents a scalable model for phishing detection. One of the largest publicly available datasets with over six million legitimate and phishing URLs has been used to test our proposed model in this study. In contrast to previous studies that rely solely on imbalanced and small datasets, the large-scale datasets used in our study ensure broad generalization to real-world scenarios. The key contributions of this study can

be summarized as follows. *Firstly*, we introduce a novel PhishFusionNet model that combines raw character-level embeddings processed by convolutional layers with handcrafted statistical URL features. Unlike traditional or single-branch architectures, this dual-branch architecture enables the model to simultaneously capture semantic URL patterns and structural, domain-specific cues, yielding a more discriminative and richer representation. *Secondly*, to maximize the model's efficacy, we applied advanced training methods, including optimized training schedules, systematic hyperparameter tuning, and regularization strategies. These improvements significantly minimize overfitting while keeping generalization over unseen URLs. *Lastly*, to evaluate our proposed model's relative effectiveness, we performed a comprehensive comparison between the PhishFusionNet model and several ML and DL algorithms. Our proposed model consistently outperforms competing models.

This paper is organized as follows: the introduction appears in Section 1, the literature and other related work are reviewed in Section 2. Section 3, meanwhile, presents the apparatus and equipment employed in this study. In Section 4, the proposed system in the aforementioned study is described and demonstrated. Finally, the experimental results are presented in Section 5 followed by discussions in Section 6. The summary of the study and its conclusion, limitations, and future work are introduced at the end of this paper.

2. Literature review

Since identifying phishing sites is critical for preventing attacks, numerous methods have been proposed to develop a trustworthy and efficient model. However, the ideal framework remains elusive. Therefore, in this section, we focus on DL methods, given their recent notable results.

K o r k m a z et al. [14] presented a hybrid phishing detection model that merges URL-based features and content-based analysis. The proposed model produced a content-based phishing-detection dataset and a high-risk URL dataset, both sourced solely from suspicious entries in PhishTank. This hybrid model achieved an accuracy of 98.37%, demonstrating the effectiveness of combining content data and URL features to enhance phishing detection in real-world scenarios. D a w a b s h e h et al. [15] developed an improved deep learning model for phishing detection that applied directly to URLs. Also, they used an extra mechanism, such as external APIs and blacklists, in order to improve performance and decrease detection time, and their results showed a high degree of accuracy in detecting phishing attacks. Z h u et al. [16] presented a CCBLA (char-convolutional and BiLSTM with attention mechanism), a lightweight deep learning model to automatically extract discriminative attributes directly from raw URLs. Two benchmark datasets were used to test their proposed model, which demonstrated strong performance in real-world implementations of phishing detection. H u s s a i n et al. [17] used a parallel one-layer CNN variant with varying kernel sizes to develop a lightweight phishing URL detection model based on a character-level. The proposed model was tested against adversarial "Offensive AI" attacks

and five publicly accessible datasets. It has good generalization across all tested datasets and shows a resistance to hostile attacks with over 99% accuracy. Opara, Chen and Wei [18] used raw URL sequences and HTML content features to develop an end-to-end deep neural network to identify phishing websites, namely the WebPhish model. WebPhish outperformed baseline feature-engineered methods by learn semantic dependencies from several input sources at once with an accuracy of 98.1%. Prasad and Dondeti [19] developed a PDSMV3-DCRNN framework, an ensemble deep learning framework for phishing detection that combines the Binary Grey Goose Optimization Algorithm to eliminate unnecessary features and keep the informative ones, and a Conditional Wasserstein Generative Adversarial Network (CWGAN) to solve dataset imbalance. PDSMV3-DCRNN model outperformed benchmark models with fewer features and an accuracy of 99.21.

In a recent study, Nayak, Muniyal and Belavagi [20] developed a model for phishing detection that integrates deep and wide learning with feature extraction. Their work focused primarily on reducing the dimensionality of manually engineered URL features (from 111 to 14-20) and using them with deep learning to improve detection accuracy. Although this method demonstrated the advantages of hybrid architectures, it was applied only to a small dataset of approximately 58,000 URLs. In contrast, our work uses a much larger dataset of over six million real-world URLs, retains all 40 extracted features without dimensionality reduction, and combines them with character-level URL encodings via convolutional neural networks to address the scalability and robustness of wide and deep learning. Furthermore, we provide a comprehensive performance evaluation by comparing our proposed Wide & Deep CNN with several deep neural architectures and traditional machine learning models. The incorporation of extensive training and a wider experimental scope sets our research apart and displays how applicable it is to actual phishing detection systems.

3. Related methods

The techniques used to experimentally evaluate and test the suggested architecture are described in this section.

3.1. Baseline classical models

Logistic Regression. Logistic Regression is a linear classification algorithm that uses the logistic function to estimate the probability of an instance belonging to a class [21]. It was employed as a baseline due to its simplicity, computational efficiency, interpretability, and efficiency in handling large-scale datasets with binary classification tasks. Its coefficients directly indicate the influence of each feature, which clearly shows how each feature affects the likelihood of phishing. Nevertheless, the disadvantage of this model is its inability to capture nonlinear and complex feature interactions, which reduces its ability to discover a sophisticated phishing pattern. Also, it struggles with imbalanced or high-dimensional datasets because it relies on manual feature engineering [22].

Logistic regression makes the following mathematical assumptions, given the feature vector $x = [x_1, x_2, \ldots, x_m]$:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_m x_m$$
$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_m x_m.$$

Next, the expected probability $p(x)$ of x (i.e., Pr (y=1 •x)) is determined by

$$p(x) = \sigma(z) = 1/1 + e - z_1,$$

where the sigmoid (logistic) function is represented by $\sigma(\cdot)$. The "logit", or odds and log-odds form, is

$$\log \frac{p(x)}{1} - p(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_m x_m.$$

The log-odds are a linear function of the input features, as demonstrated by this "log" formulation. Maximum Likelihood Estimation (MLE) is used in logistic regression to learn the parameters β_0, β_1,…,. β_m, The values of β are selected to maximize the likelihood of the observed class labels given the inputs.

Random Forest. Random Forest is an ensemble learning technique that builds multiple decision trees and uses majority voting to aggregate their predictions. Its key idea is to reduce variance and capture nonlinear relationships, making it a powerful baseline for identifying intricate phishing patterns in URL features [23]. A bootstrap sample of the training data sampling with replacement, is used to train each tree. In addition, only a random subset of features is considered for the optimal split when splitting a node in a tree. This feature randomness mechanism assists in the tree decorrelation and supports diversity over the ensemble [23].

Each tree votes during classification to label a class, and the final prediction of the forest is defined by the class obtaining the majority number of votes [24]. Let k be an ensemble decision tree that is denoted by $\{T_1, T_2, \ldots, T_K\}$, and each tree T_i returns a class label $h_i(x)$ for an input example x. The final decision (x) of the Random Forest is given by the best voting

$$H(x) = \arg\max_c \sum_{i=1}^{k} 1\{h_i(x) = c\},$$

where $1\{\cdot\}$ is the indicator function, and the probability of class c is estimated in probabilistic terms as follows:

$$P(c \mid x) = \frac{1}{k} \sum_{i=1}^{k} 1\{h_i(x) = c\}.$$

The Random Forest can be generalized and analyzed in terms of inter-tree correlation and the strength of individual trees. It attains high generalization and accuracy while keeping resilience to overfitting, noise, and outliers. Despite these advantages, it needs an in-depth computational memory and resources; also, it lacks transparency in its overall decision process, which complicates interpretability [25].

XGBoost. XGBoost is an advanced application of the gradient boosting algorithm that enhances the speed and scalability, which sequentially improve decision trees where each tree refines the residual of the prior one. Its ability to handle class imbalance by regularization mechanisms makes it a competitive baseline in cybersecurity-related categorization tasks [26]. The model prediction for an instance x is

$$\hat{y} = \sum_{k=1}^{k} f_k(x),$$

where k is represented by each f_k •F, and a regression tree is added in k iteration. With regularization, the training goal is

$$\min_{\{f_k\}} \left[\sum_{i=1}^{n} L_{(y_i, y^i)} + \sum_{k+1}^{k} \Omega_{f(k)}\right],$$

where L is the loss function (such as logistic loss) and $\Omega(f)=\gamma T+12\lambda^1 w^2{}_2$ is a regularization term that relies on the number and weights of leaves. XGBoost uses a second-order Taylor expansion of the loss (such as gradients and Hessians) to fit each new tree to estimated residuals. XGBoost provides efficient, scalable, and highly accurate learning by integrating boosting and regularization techniques, which increases its capability in capturing nonlinear feature interactions. However, it requires extensive hyperparameter tuning and incurs a high computational expense for large ensembles [26].

3.2. Deep learning models

Convolutional Neural Network (CNN). CNN has been widely used for sequence modeling tasks such as text and URL analysis despite its original introduction for image and signal recognition [27]. The CNN architecture was typically from convolutional, pooling, and fully connected layers that sequentially map the output class from inputs.

CNN's main operation involves the application of learnable filters that go over the input to execute convolutional operations. Each filter executes the dot product between a local patch of the input and its weights to introduce a feature map that focuses on the spatial distribution of learned patterns. The design principle of local connectivity and shared weights drastically minimizes the number of parameters compared to dense architectures, leading to improved parameter efficiency and generalization [29]. After convolution, the network applies a nonlinear activation function (such as ReLU) element-by-element to model complex nonlinear relationships. The following pooling layers (such as max pooling) lower spatial dimensions by summarizing local regions, making the model invariant to small translations and lowering computational cost [27].

In the end, the feature maps are global pooled (flattened) and go through fully connected (dense) layers, terminating with an output layer (e.g., a sigmoid or softmax activation) that produces probability scores for each class. CNNs optimize their filters and weights via backpropagation and gradient descent during training, bringing the learned representations with the classification goal [28]. CNNs are particularly effective for phishing URL analysis and automatically extract both local and hierarchical patterns from raw input (such as character sequences) without the need for human feature engineering, and parameter sharing enables them to generalize effectively with fewer parameters.

The proposed CNN model (illustrated in Fig. 1) learns low-dimensional semantic representations using an embedding layer that maps character-level URL tokens to 64-dimensional vectors. The first convolutional layer applies multiple 1D filters to capture local sequential patterns such as odd token arrangements or suspicious substrings. By gradually reducing the feature map dimension, the pooling layers improve translation invariance and minimize overfitting. Finally, the flattened features have been run through a fully connected layer with dropout

regularization (rate = 0.5) to avoid overfitting, followed by a sigmoid activation to output the phishing probability.

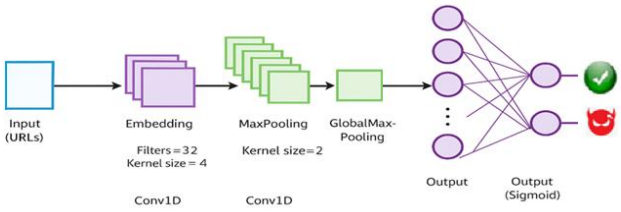


Fig. 1. Architecture of the proposed CNN model

LSTM+CNN. LSTM+CNN is a hybrid model that combines the strengths of Long Short-Term Memory (LSTM) units with CNN layers [29]. This hybrid model enables the architecture to concurrently capture local dependencies using convolutions and long-range sequential dependencies using LSTMs. Because of this, it works effectively for modeling URL sequences where malicious intent may be embedded in both short- and long-range structures [29]. The hybrid design enables the model to identify both local structural anomalies (via CNN) and long-range sequential dependencies (via LSTM), and provides more thorough feature extraction than either one alone. Nevertheless, the combined architecture increases model complexity and computational cost, and the performance depends on balancing between convolutional and recurrent components. Fig. 2 depicts the architecture of the proposed model.

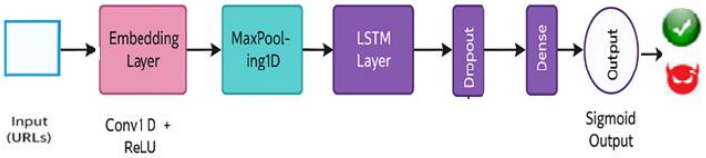


Fig. 2. Architecture of the proposed LSTM+CNN model

The model begins with an Embedding Layer that converts each URL's character-level representation into dense, low-dimensional vectors to encode syntactic and semantic relationships among characters. After that, the embeddings are then passed through a 1D Convolutional Layer (Conv1D) that employs a number of filters to detect local n-gram patterns (such as "http", "login", "secure") that are commonly found in phishing URLs. The MaxPooling1D Layer is then applied after the convolutional layer to reduce computational complexity, downsample the feature maps, and highlight the most salient patterns. Afterwards, the extracted local features are fed into an LSTM Layer, which models sequential dependencies throughout the entire URL string. This enables the network to store contextual information that may indicate phishing intent, such as patterns at the beginning or end of the URL. The dropout layer is used during training to prevent overfitting by randomly disabling a portion of neurons. Finally, the probability of whether the URL is legitimate or phishing is determined by the Sigmoid Activation function that follows the Fully Connected (Dense) Layer.

CNN with attention. An attention mechanism was added to the CNN model to increase interpretability and highlight critical subsequences [30]. The attention layer enables the network to reduce the weight of irrelevant segments, dynamically assign weights to different URL segments, and focus more on suspicious substrings. By adding the attention mechanism to the CNN layers, the convolutional architecture of the network will be selectively focused on the most instructive segments of the URL sequence [30]. The attention weights α_i are learned mathematically from a feature sequence h=[h_1, h_2,…, h_n] generated by the CNN:

$$\alpha_i = \frac{\exp(\tanh(W_h h_i + b))}{\sum_j \exp(\tanh(W_h h_i + b))},$$

and the context vector is calculated as

$$c = \sum_i \alpha_i h_i,$$

where b and W_h are trainable parameters. The aggregated, weighted data sent to the classifier is captured by the context vector c.

Fig. 3 illustrates the overall architecture, with the attention layer dynamically weighting the local features according to their relevance to phishing detection, while the convolutional layers extract features.

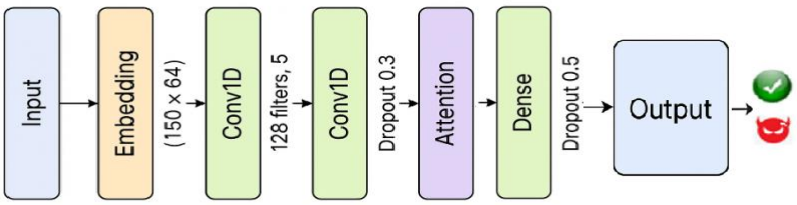


Fig. 3. Architecture of the proposed CNN with attention mechanism

In our study, we utilized character-level features extracted from raw URL strings to train the CNN+Attention model for phishing URL detection. Each URL was encoded to a 150-character fixed-length sequence that was then passed through a 64-dimensional dense representation embedding layer. To extract spatial patterns and local dependencies from the structure of the URL, one-dimensional convolutional layers (Conv1D) with ReLU activation were utilized to process the embedded sequences. Convolutional feature maps were then assigned weights to different URL regions and highlighted the most informative subsequences, such as special symbols, domain names, and doubtful substrings frequently found in phishing links. To reduce overfitting, the dropout regularization layer was applied with a rate of 0.4 after the attention-weighted context vector was flattened. Then the sigmoid activation function is applied to categorize the URL as either legitimate or phishing.

4. Experimental methodology

This section describes the architecture of the proposed PhishFusionNet model, then gives a clear description of the dataset used for evaluation. Then end with describing the evaluation criteria used to determine the model's effectiveness.

4.1. The proposed PhishFusionNet architecture

The objective of this study is to propose the novel PhishFusionNet model that integrates both handcrafted features and raw URL character-level patterns into a unified and cohesive framework (refer to Fig. 4 and Table 1).

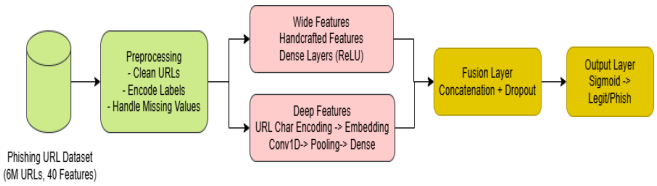


Fig. 4. The PhishFusionNet model's architecture

The PhishFusionNet architecture comprises two parallel components. The first part is the wide path. This part is responsible for handling the handcrafted features that were extracted from the dataset. The extracted features, such as the number of dots, the existence of special characters, and the length of the URL, are then fed to the fully connected dense layers with ReLU activation functions. This part of the model was designed to catch both linear and nonlinear relationships found in manually engineered features. The model's second part is the deep path, which is responsible for learning patterns directly from raw URL strings. Where each character extracted from the URL sequence is embedded into a dense vector to provide a more continuous and richer feature space. Then, to identify local sequential dependencies, such as malicious *n*-gram patterns or suspicious substrings, the embedded sequence is applied across the Convolutional layers (Conv1D). Finally, the most important features that are extracted from the convolutional layer pass through a global max pooling layer, and are then fed into the dense layer to set up higher-level abstract representations.

The fusion layer represents the integration point between the handcrafted features (wide part) and automatically learned features (deep part). At this point, dropout regularization is used to prevent overfitting and improve the model's capacity to generalize to unseen data. Finally, the fused representation is directed into the output layer, a dense neuron with sigmoid activation, which assigns each input URL a likelihood score between zero and one. The model classifies the URL as either legitimate or phishing based on a threshold, normally set at 0.5. The PhishFusionNet model's pseudo-code (Algorithm 1) is shown below.

Algorithm 1. PhishFusionNet Model (Wide and Deep Hybrid URL Classification Model)

Step 1. *Input:* Raw URL Dataset *D*, Handcrafted Features *F*, Vocabulary *V*, Max Length *L*

Step 2. *Output:* Trained PhishFusionNet model

Step 3. Initialize deep_input ← Encoded URLs ($L \times 1$)

Step 4. Initialize wide_input ← Extracted handcrafted features *F*

Step 5. Initialize Xdeep ← Character-level deep branch input

Step 6. Initialize Xwide ← Wide branch input

Step 7. for each URL ∈ *D* do

Step 8. Preprocess URL (lowercase, clean, tokenize)

Step 9. Encode URL using vocabulary *V* with max length *L*
Step 10. Append encoded URL to *X*deep
Step 11. Extract handcrafted lexical & statistical features
Step 12. Append extracted features to *X*wide
Step 13. end for
Step 14. Build Deep Branch
Step 15. embedding ← Embedding(*X*deep, embedding_dim = 64)
Step 16. conv ← Conv1D(filters = 128, kernel_size = 5)(embedding)
Step 17. deep_repr ← GlobalMaxPooling1D(conv)
Step 18. deep_repr ← Dense(64, activation = ReLU)(deep_repr)
Step 19. Build Wide Branch
Step 20. wide_repr ← Dense(64, activation = ReLU)(*X*wide)
Step 21. Fusion Layer
Step 22. *Z* ← Concatenate([deep_repr, wide_repr])
Step 23. Z ← Dropout(rate = 0.5)(*Z*)
Step 24. Z ← Dense(64, activation = ReLU)(*Z*)
Step 25. Output Layer
Step 26. $\hat{y}$ ← Dense(1, activation = Sigmoid)(*Z*)
Step 27. Model Training
Step 28. Compile model using optimizer Adam and loss Binary Crossentropy
Step 29. Train model on (*X*deep, *X*wide) with batch size *B* for *E* epochs
Step 30. Evaluation
Step 31. Compute Accuracy, Precision, Recall, F1-score, AUC
Step 32. Return Trained PhishFusionNet model

This hybrid framework strengthens the model and allows it to take advantage of both deep representation learning and traditional feature engineering. Where the wide path capitalizes on domain knowledge encoded into handcrafted features, while the deep path comes across hidden sequential patterns directly from the URL strings. By merging these two complementary techniques, the proposed framework surpasses both the conventional machine learning and the deep learning baselines.

Table 1. Parameters of the PhishFusionNet model

Layer (Type)	Output shape	Parameters	Input source / Description
Wide Input (Input Layer)	(None, 37)	0	Handcrafted features (scaled, normalized)
Dense (Wide Path)	(None, 64)	2432	Fully connected layer with ReLU activation
Deep Input (Input Layer)	(None, 150)	0	Character-level encoded URL sequence
Embedding	(None, 150, 64)	14,912	Character embedding with a 64-dim vector space
Conv1D	(None, 146, 128)	41,088	1D convolution with 128 filters and kernel size = 5
Global Max Pooling	(None, 128)	0	Extracts maximum activation from each feature map
Dense (Deep Path)	(None, 64)	8256	Fully connected layer with ReLU activation
Concatenate	(None, 128)	0	Combines outputs of wide and deep branches
Dropout	(None, 128)	0	Dropout rate = 0.4 to prevent overfitting
Dense	(None, 64)	8256	ReLU activation for feature fusion
Output layer (dense)	(None, 1)	65	Sigmoid activation for binary classification (Legitimate / Phishing)
Total parameters	75,009 (≈293 KB)		

4.2. Dataset description

This study utilizes a large-scale real-world URL dataset obtained from a publicly available Kaggle repository introduced by M a h a j a n [31]. The dataset consists of 6,093,263 URLs labeled as either legitimate (0) or phishing (1), as summarized in Table 2. The dataset has an equitable distribution, with 56.2% of these entries being legitimate and 43.8% being phishy to ensure a rigorous evaluation of both classes throughout classification [31].

Table 2. Dataset description

Dataset	Total instances	Legitimate	Phishing	Features
Kaggle dataset	6,093,263	3,422,269	2,670,994	40

Every sample of the dataset contains 40 engineered features that give details about multiple facets of the URL's construction, and these attributes were taken from the URL string's lexical, statistical, and structural characteristics.

4.3. Dataset preprocessing

To guarantee the quality and consistency before model training, the dataset goes through a series of preprocessing steps. First, all duplicates and malformed entries of the URLs were eliminated, and then the labels were encoded into binary format, and the legitimate instances were set to 0, and the phishing instances were set to 1. In order to avoid learning bias, missing values are handled properly. Furthermore, to keep the handcrafted features on a comparable scale, z-score normalization is also used to standardize them.

For the wide learning component, all manually created features, including URL length, ratio of digits to characters, presence of HTTPS, number of subdomains, and other lexical statistics, were extracted. These features, which are frequently used in phishing detection research, capture the structural and statistical properties of URLs. Then, to prevent features with larger numerical ranges from controlling the learning process, these features were normalized using z-score standardization to ensure comparability over variables. For the deep learning part, raw URLs were transformed into fixed-length sequences using a character-level encoding scheme. First, all of the distinct characters in the dataset were used to create a vocabulary, and each character was assigned a distinct integer index. Following that, all URLs were padded and truncated to a uniform length of 150 characters. Because of this transformation step, the deep branch of the proposed model was able to capture fine-grained lexical patterns that are frequently used as indicative of phishing attempts. Finally, the dataset was divided into 80% for the training and 20% for the testing process using stratified sampling.

4.4. Evaluation criteria

The proposed model was evaluated using several regularly used metrics, including Accuracy, Precision, Recall, and the F1-score. These assessment tools offer a comprehensive evaluation of the model predictive. Where the accuracy represents the general indicator that quantifies the percentage of correctly classified instances among the overall samples. Precision measures the proportion of all predicted

positives to the correctly predicted positive cases. Recall estimates the percentage of real positive cases that the model correctly recognizes. Finally, the F1-score is a harmonic mean that balances the trade-off between Precision and Recall and is particularly useful when the dataset is imbalanced

$$\text{Accuracy} = \frac{\text{TC}}{\text{TC+FC}},$$

where TC and FC represent the number of correctly and incorrectly classified instances, respectively,

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}, \text{Recall} = \frac{\text{TP}}{\text{TP+FN}},$$

where TP, FP, and FN are True Positive, False Positive, and False Negative, respectively.

5. Experimental results and discussion

Extensive experiments were carried out in this study to evaluate the power of the proposed model's ability to detect phishing attacks. It compared it with several deep learning and traditional machine learning models to establish a comprehensive benchmark. Deep learning models are implemented using Keras with a TensorFlow backend, and the machine learning algorithms are implemented using the Scikit-Learn library.

5.1. Results of the traditional machine learning algorithms

The main goal of this study is to design a model for phishing detection; however, to establish a baseline performance benchmark, the dataset was also assessed using traditional machine learning algorithms. This assessment provides a useful benchmark for evaluating the limitations and strengths of traditional ML methods relative to the proposed model. Table 3 presents the results of the traditional ML models, where the Logistic Regression reached an accuracy of 88.29%, along with a precision of 85.08%, F1-score of 86.93%, and recall of 88.86%. Random Forest significantly obtained better results with an accuracy of 95.28%, a recall of 94.26%, a precision of 94.93%, and an F1-score of 94.59%. In addition, a competitive performance was shown by XGBoost with an accuracy of 93.64%, precision 92.60%, recall 92.92%, and F1-score 92.76%. These results, especially from ensemble models like Random Forest and XGBoost, show that handcrafted URL features can be effectively leveraged for phishing detection using traditional machine learning models.

Table 3. Results of the Traditional Machine Learning Algorithms

Algorithm	Accuracy	Precision	Recall	F1-score
Logistic Regression	0.8829	0.8508	0.8886	0.8693
Random Forest	**0.9528**	**0.9493**	**0.9426**	**0.9459**
XGBClassifier	0.9364	0.926	0.9292	0.9276

Even with the insights provided by accuracy, precision, recall, and F1-score, confusion matrices are still an important tool that enables us to directly examine the distribution of correct and incorrect classes across phishing and legitimate URLs. Fig. 5 shows the confusion matrices obtained for the Random Forest, XGBoost, and

Logistic Regression classifiers. The confusion matrices make it easier to recognize whether a model tends to favor one class over another, which is important for phishing detection because false negatives present a significant security risk.

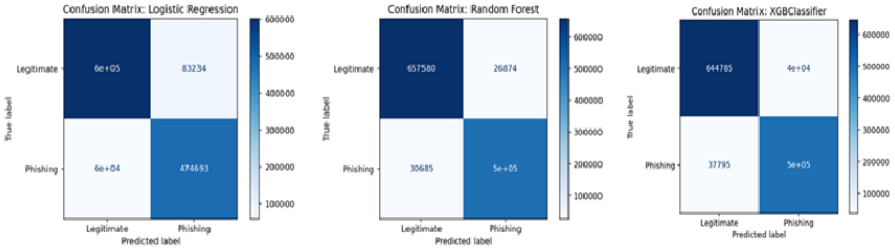


Fig. 5. Confusion matrix of traditional machine learning algorithms

5.2. Results of the deep learning models

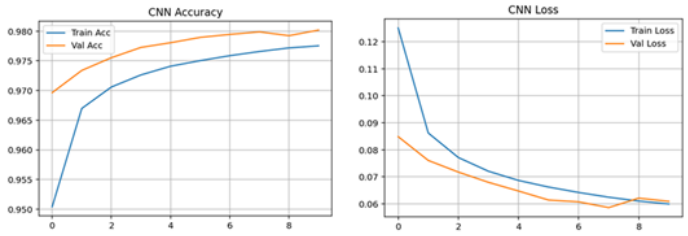
Unlike ML methods that rely entirely on manual feature engineering, DL models can automatically learn abstract and hierarchical representations from raw attributes. DL is an encouraging method for phishing detection and has shown remarkable success in tasks like cybersecurity. Three DL models were used in this study: *First* a CNN model that captures sequential and spatial patterns in URLs, *second* a hybrid LSTM+CNN model that merges the feature extraction capabilities of the CNN model with the power of LSTM for sequence modeling, and *finally* a CNN with Attention model was used to highlight the most informative URL subsequences by using a CNN that augments convolutional layers with an attention mechanism.

Table 4 displays the results of the deep learning models in terms of accuracy, precision, recall, and F1-score. The LSTM+CNN architecture achieved the highest performance with an accuracy of 98.4%. Followed by the CNN model with an accuracy of 98.0%. Finally, the CNN+Attention model also yielded competitive results with an accuracy of 97.66%.

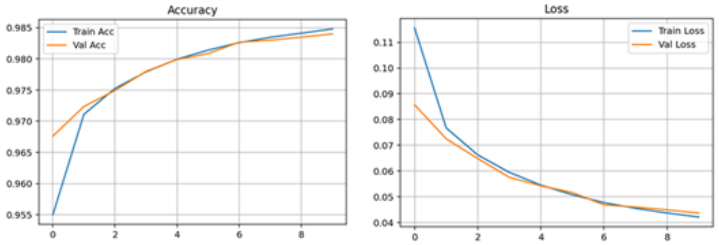
Table 4. Results of the Deep Learning Models

Algorithm	Epochs	Accuracy	Precision	Recall	F1-score
CNN	10	0.9801	0.9774	0.9773	0.9773
LSTM+CNN	**10**	**0.984**	**0.9833**	**0.9801**	**0.9817**
CNN+Attention	10	0.9766	0.9862	0.9601	0.973

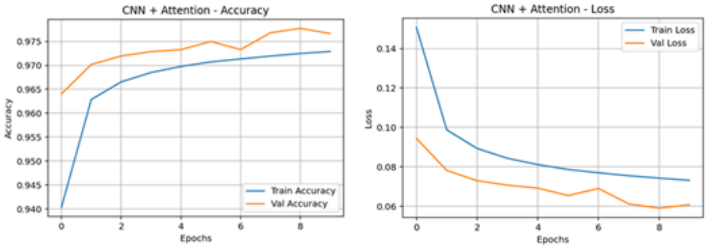
Fig. 6 presents the three models' training and validation accuracy curves, as well as the corresponding training and validation loss curves across epochs. These plots provide worthy information on the convergence behavior, generalization ability, and possible risks of overfitting or underfitting hazards. These findings reveal that the LSTM+CNN and CNN architectures have higher performance, which converges fast and maintains validation performance.



(a) Accuracy and Training Loss for CNN Model



(b) Accuracy and Training Loss for CNN+LSTM Model



(c) Accuracy and Training Loss for CNN+Attention Model

Fig. 6. Accuracy and loss for Training and Testing data of the three Deep Learning models

Fig. 7 presents the confusion matrices of the three models, where CNN and CNN with Attention show slightly different tendencies in misclassification, in contrast to the LSTM+CNN model, which shows the most balanced performance with both low false negatives and false positives. This analysis is especially essential for phishing detection, as false negatives can expose users directly to cyberattacks, while false positives may lessen usability by flagging safe websites needlessly.

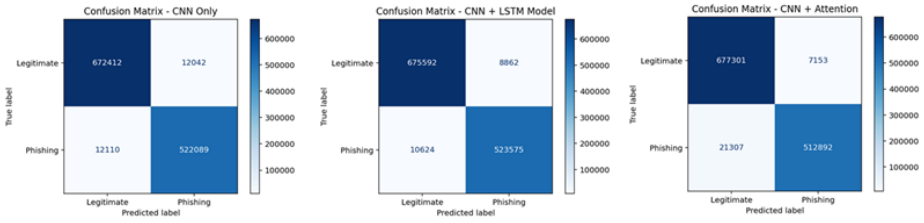


Fig. 7. Confusion matrix of Deep Learning Models

5.3. Results of the proposed PhishFusionNet Model

As shown in Table 5, the proposed model (PhishFusionNet) achieved the highest performance with an accuracy of 98.9%, precision of 0.9888, recall of 0.9861, and F1-score of 0.9875. These results notably outperform ML models, including Random Forest (95.3%) and the strongest DL models, including LSTM+CNN (98.4%).

Table 5. Results of the PhishFusionNet Model

Algorithm	Epochs	Accuracy	Precision	Recall	F1-score	ROC AUC
PhishFusionNet	50	**0.989**	0.9888	0.9861	0.9875	**0.9992**

These results show the effectiveness of the PhishFusionNet's model, which combines character-level URL sequence learning (CNN branch) with handcrafted features (wide branch), producing a model that is both precise and a highly robust framework for phishing detection in large-scale, real-world situations. Fig. 8 displays the accuracy and loss curves for training and validation over all epochs for the proposed model. These curves offer useful information about the convergence behavior and model generalization.

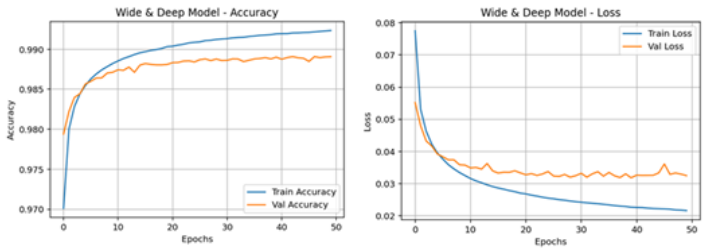


Fig. 8. Accuracy and loss for Training and Testing data of the PhishFusionNet model

As we can observe from Fig. 8, the proposed model achieved a fast convergence and high classification ability. The validation and training accuracy curves show a consistent upward trend, stabilizing at 98.9%. Also, as shown by the loss curves, the model significantly generalized without overfitting. These smooth convergence patterns confirm the robustness of the proposed architecture in learning complex legitimate and phishing URL representations. Fig. 9 shows the confusion matrix of the proposed PhishFusionNet model, where the predictions fall along the diagonal, indicating the model reliably and consistently distinguishes the phishing URLs from legitimate ones with a few misclassifications observed.

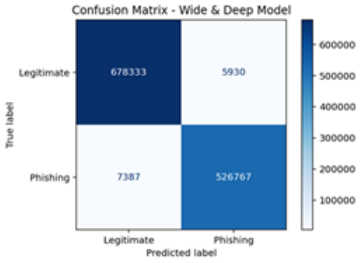


Fig. 9. Confusion matrix of the proposed PhishFusionNet model

5.4. Discussion

In this study, the experimental evaluation provides a comprehensive comparison between the proposed PhishFusionNet framework and the traditional machine learning techniques and cutting-edge deep learning architectures. The results not only show the effectiveness of the proposed model but also provide many significant insights about the restrictions of prior methods in addressing the phishing URL detection challenge.

First, the baseline ML models achieved reasonable performance, with Random Forest, which achieved the highest accuracy of 95.3% among the rest of the models, such as Logistic Regression and XGBoost. Though these models are more interpretable and efficient, their dependence on manually generated features limits their applicability to the diverse and evolving phishing URL structures. This drawback and discrepancy in performance emphasize the necessity of more elastic learning strategies that can extract complex and hidden patterns automatically. In addition, opposite to the ML models, the DL models present a notable improvement, where the CNN model achieved the highest accuracy 89.0% which indicates the strength of the DL models in capturing local sequential patterns in URLs. The performance was improved further by using a hybrid LSTM+CNN model with an accuracy that reached 98.4%, which emphasizes the importance of the integration of the local feature with the sequence model. While CNN with the Attention mechanism achieved the highest precision (98.6%), which indicates that while attention assists the model in focusing on discriminative subsequences, it may sometimes underperform in recognizing all phishing variations.

In contrast, the proposed PhishFusionNet model outperformed all ML baselines and deep learning models with an accuracy of 98.9. This superior performance of the proposed model comes from two crucial design decisions: (1) by combining wide handcrafted features with deep character-level URL embeddings, which allow the model to leverage both data-driven representations and domain knowledge; (2) minimizing overfitting and improving generalization by using regularization and hyperparameter tuning throughout the training process. The effect of the high performance of the proposed model goes beyond the numerical values, where a high recall value guarantees that phishing URLs are successfully detected, and that will effectively prevent phishing attacks. In addition, high precision ensures that legitimate URLs are not incorrectly flagged as phishy, which minimizes disruptions to users and institutions. PhishFusionNet's model flexibility in both dimensions shows that it can be deployed in real-time cybersecurity systems.

6. Conclusion

This research paper presented a comprehensive comparative analysis of DL and classical ML models for phishing URL detection, and ended with the proposed PhishFusionNet model. The best results were reached by the PhishFusionNet proposed model, which combines deep character-level embeddings with wide handcrafted features, achieving the highest performance by attaining an accuracy of

98.9%. These results emphasize that generalization and interpretability can be acquired through hybrid architectures, which can be reached by combining learned representations with explicit feature-based knowledge. This study generally highlights that the PhishFusionNet model provides a scalable and very accurate real-world phishing detection solution that can be used effectively on large-scale datasets and can be implemented in practical cybersecurity defense systems and browser-level phishing protection tools.

Limitations and future work. Even with the strong performance of the proposed PhishFusionNet model, the model mainly depends on static URL-based and lexical features. This might make it harder for the model to adjust to dynamic phishing techniques. To improve robustness in the future can be expanded to incorporate behavioral and real-time content analysis as well as integrating visual, HTML, and URL features. Also, despite the Large Language Models (LLMs) demonstrating a promising capacity in cybersecurity-related text analysis, their high computational cost and inference latency restrict their feasibility for large-scale, real-time phishing URL detection. Future work will explore the possibility of integrating a lightweight or distilled LLM-based approach into phishing detection frameworks. Finally, the explainability mechanisms can be added to increase openness and strengthen confidence in automated phishing detection systems, and allow cybersecurity professionals to interpret model judgments.

References

1. Perwej, Y., S. Q. Abbas, J. P. Dixit, N. Akhtar, A. K. Jaiswal. A Systematic Literature Review on the Cyber Security. – International Journal of Scientific Research and Management, Vol. **9**, 2021, No 12, pp. 669-710.
2. Veprytska, O., V. Kharchenko, O. Illiashenko. Cybersecurity and Artificial Intelligence: Triad-Based Analysis and Attacks Review. – Cybernetics and Information Technologies, Vol. **25**, 2025, No 3, pp. 156-185.
3. Anti-Phishing Working Group (APWG). Phishing Activity Trends Report, 1st Quarter 2025. APWG, 2025.
4. Basit, A., M. Zafar, X. Liu, A. R. Javed, Z. Jalil, K. Kifayat. A Comprehensive Survey of AI-Enabled Phishing Attack Detection Techniques. – Telecommunication Systems, Vol. **76**, 2021, No 1, pp. 139-154.
5. Zonyfar, C., J.-B. Lee, J.-D. Kim. HCNN-LSTM: Hybrid Convolutional Neural Network with Long Short-Term Memory Integrated for Legitimate Web Prediction. – Journal of Web Engineering, Vol. **22**, 2023, No 5, pp. 757-782.
6. Shirazi, H., S. R. Muramudalige, I. Ray, A. P. Jayasumana, H. Wang. Adversarial Autoencoder Data Synthesis for Enhancing Machine Learning-Based Phishing Detection Algorithms. – IEEE Transactions on Services Computing, Vol. **16**, 2023, No 4, pp. 2411-2422.
7. Soni, U., A. G. Jethava. Latest Advancements in Credit Risk Assessment with Machine Learning and Deep Learning Techniques. – Cybernetics and Information Technologies, Vol. **24**, 2024, No 4, pp. 22-44.
8. Priya, S., S. Selvakumar, R. L. Velusamy. PaSOFuAC: Particle Swarm Optimization-Based Fuzzy Associative Classifier for Detecting Phishing Websites. – Wireless Personal Communications, Vol. **125**, 2022, No 1, pp. 755-784.
9. Nowroozi, E., M. Mohammadi, M. Conti. An Adversarial Attack Analysis on a Malicious Advertisement URL Detection Framework. – IEEE Transactions on Network and Service Management, Vol. **20**, 2022, No 2, pp. 1332-1344.

10. F e n g, T., C. Y u e. Visualizing and Interpreting RNN Models in URL-Based Phishing Detection. – In: Proc. of 25th ACM Symposium on Access Control Models and Technologies, ACM, 2020, pp. 13-24.
11. B a h n s e n, A. C., E. C. B o h o r q u e z, S. V i l l e g a s, J. V a r g a s, F. A. G o n z á l e z. Classifying Phishing URLs Using Recurrent Neural Networks. – In: Proc. of 2017 APWG Symposium on Electronic Crime Research (eCrime), IEEE, 2017, pp. 1-8.
12. Z h a n g, X., J. Z h a o, Y. L e C u n. Character-Level Convolutional Networks for Text Classification. – Advances in Neural Information Processing Systems, Vol. **28**, 2015.
13. O p a r a, C., B. W e i, Y. C h e n. HTMLPhish: Enabling Phishing Web Page Detection by Applying Deep Learning Techniques on HTML Analysis. – In: Proc. of International Joint Conference on Neural Networks (IJCNN'20), IEEE, 2020, pp. 1-8.
14. K o r k m a z, M., E. K o c y i g i t, O. S a h i n g o z, B. D i r i. A Hybrid Phishing Detection System Using Deep Learning-Based URL and Content Analysis. – Elektronika ir Elektrotechnika, Vol. **28**, 2022, No 5.
15. D a w a b s h e h, A., M. J a z z a r, A. E l e y a n, T. B e j a o u i, S. P o p o o l a. An Enhanced Phishing Detection Tool Using Deep Learning from URL. – In: Proc. of International Conference on Smart Applications, Communications and Networking (SmartNets'22), IEEE, 2022, pp. 1-6.
16. Z h u, E., Q. Y u a n, Z. C h e n, X. L i, X. F a n g. CCBLA: A Lightweight Phishing Detection Model Based on CNN, BiLSTM, and Attention Mechanism. – Cognitive Computation, Vol. **15**, 2023, No 4, pp. 1320-1333.
17. H u s s a i n, M., C. C h e n g, R. X u, M. A f z a l. CNN-Fusion: An Effective and Lightweight Phishing Detection Method Based on a Multi-Variant ConvNet. – Information Sciences, Vol. **631**, 2023, pp. 328-345.
18. O p a r a, C., Y. C h e n, B. W e i. Look Before You Leap: Detecting Phishing Web Pages by Exploiting Raw URL and HTML Characteristics. – Expert Systems with Applications, Vol. **236**, 2024, 121183.
19. P r a s a d, Y. B., V. D o n d e t i. PDSMV3-DCRNN: A Novel Ensemble Deep Learning Framework for Enhancing Phishing Detection and URL Extraction. – Computers & Security, Vol. **148**, 2025, 104123.
20. N a y a k, G. S., B. M u n i y a l, M. C. B e l a v a g i. Enhancing Phishing Detection: A Machine Learning Approach with Feature Selection and Deep Learning Models. – IEEE Access, 2025.
21. G e n g, Y., Q. L i, G. Y a n g, W. Q i u. Logistic Regression. – In: Practical Machine Learning Illustrated with KNIME, Springer, 2024, pp. 99-132.
22. Z o u, X., Y. H u, Z. T i a n, K. S h e n. Logistic Regression Model Optimization and Case Analysis. – In: Proc. of 7th IEEE International Conference on Computer Science and Network Technology (ICCSNT'19), IEEE, 2019, pp. 135-139.
23. S u n, Z., G. W a n g, P. L i, H. W a n g, M. Z h a n g, X. L i a n g. An Improved Random Forest Based on the Classification Accuracy and Correlation Measurement of Decision Trees. – Expert Systems with Applications, Vol. **237**, 2024, 121549.
24. Z h u, E., Z. C h e n, J. C u i, H. Z h o n g. MOE/RF: A Novel Phishing Detection Model Based on a Revised Multiobjective Evolution Optimization Algorithm and Random Forest. – IEEE Transactions on Network and Service Management, Vol. **19**, 2022, No 4, pp. 4461-4478.
25. S a l m a n, H. A., A. K a l a k e c h, A. S t e i t i. Random Forest Algorithm Overview. – Babylonian Journal of Machine Learning, Vol. **2024**, 2024, pp. 69-79.
26. I k r a m, S. T., A. K. C h e r u k u r i, B. P o o r v a, P. S. U s h a s r e e, C. Z h a n g, X. L i u, G. L i. Anomaly Detection Using XGBoost Ensemble of Deep Neural Network Models. – Cybernetics and Information Technologies, Vol. **21**, 2021, No 3, pp. 175-188.
27. I g e, A. O., M. S i b i y a. State-of-the-Art in 1D Convolutional Neural Networks: A Survey. – IEEE Access, 2024.
28. M a'a r i f, A., W. R a h m a n i a r, H. I. K. F a t h u r r a h m a n, A. Z. K. F r i s k y. Understanding of Convolutional Neural Network (CNN): A Review. – International Journal of Robotics & Control Systems, Vol. **2**, 2022, No 4.
29. A l s h i n g i t i, Z., R. A l a q e l, J. Al-Muhtadi, Q. E. U. H a q, K. S a l e e m, M. H. F a h e e m. A Deep Learning-Based Phishing Detection System Using CNN, LSTM, and LSTM-CNN. – Electronics, Vol. **12**, 2023, No 1, 232.

30. S o y d a n e r, D. Attention Mechanism in Neural Networks: Where it Comes and Where it Goes. – Neural Computing and Applications, Vol. **34**, 2022, No 16, pp. 13371-13385.
31. M a h a j a n, S. 6.5 Lakh URLs Labelled as 1 and 0. Kaggle Dataset. **https://www.kaggle.com/datasets/somanshumahajan/65-lakh-urls-labelled-as-1-and-0/data**.

Received: 17.12.202, Revised version: 08.02.2026, Accepted: 14.02.2026