*electronics*

MDPI

*Article*

# PLG-URLNet: Layer-Wise Fusion of Local Convolution and Global Attention for Detecting Malicious URLs

**Yongcai Xiao [1], Yongping Liu [2], Yueshun He [3,*], Linlin He [2], Heng Liu [1], Yuxin Deng [1], Shuhui Pan [1] and Hongzhi Liu [1]**

[1] State Grid Jiangxi Electric Power Research Institute, State Grid Corporation of China, Nanchang 330096, China
[2] School of Software, East China University of Technology, Nanchang 330013, China
[3] School of Artificial Intelligence and Information Engineering, East China University of Technology, Nanchang 330013, China
* Correspondence: heys@ecut.edu.cn; Tel.: +86-18324135059

**Abstract**

Malicious URLs exhibit diverse patterns and rapid evolution in complex network environments. Traditional detection methods often focus solely on local patterns or global dependencies, failing to capture multi-scale information effectively, thus limiting their performance on obfuscated characters and structural variants. To address these limitations, this paper proposes PLG-URLNet, a malicious URL detection method based on layer-by-layer interaction of local and global features. The model adopts a two-branch structure: a convolutional branch extracts local patterns and contextual information, while a Transformer branch models global dependencies and long-range relationships. A hierarchical interaction module (PLGIM) embedded in each layer enables effective feature integration, enhancing representation capabilities progressively. A lightweight decoder-type fusion layer is further introduced for final discrimination. To avoid over-estimating generalization ability caused by random train–test splitting, both standard evaluation protocols and time-sequential partitioning are adopted. Experiments show that PLG-URLNet achieves an accuracy of 98.59%, maintaining performance advantages in stringent temporal generalization scenarios and effectively identifying obfuscated and variant URLs. This method provides a feasible solution for malicious URL detection.

**Keywords:** malicious URL detection; feature interaction; convolutional neural network (CNN); transformer; attention mechanism

## 1. Introduction

With the rapid development of the Internet, network attack techniques have continued to evolve. Malicious URLs have become the primary vector for phishing, online fraud, and malware distribution, posing severe threats to both individual users and organizational cybersecurity [1]. Studies and survey reports indicate that the scale of phishing and malicious link attacks has kept rising in recent years. The attack surface has expanded from domain names and paths to request parameters, making detection increasingly challenging [2]. Meanwhile, attackers frequently employ obfuscation strategies such as homograph domains, typo-squatting, and character-level perturbations (e.g., mixed alphanumeric symbols) to evade static detection mechanisms [3,4]. Therefore, enhancing intelligent detection and protection capabilities against malicious URLs has become an urgent demand in network security.

Check for updates

Academic Editor: Stefanos Kollias

Received: 22 April 2026
Revised: 22 May 2026
Accepted: 26 May 2026
Published: 5 June 2026

**Copyright:** © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Early mainstream malicious URL detection mainly relies on blacklist–whitelist matching, which is simple and widely adopted in phishing website defense [5]. The core idea is to compare a target URL with predefined blacklists before user access and block known phishing pages [6]. This mechanism is widely used in modern browsers; for instance, Google Safe Browsing maintains frequently updated blacklists to intercept suspicious sites and provides APIs for developers to perform pre-visit checks [7]. However, phishing websites feature fast generation and short lifecycles. Blacklists require frequent updates yet still offer limited coverage, failing to respond to zero-day or rapidly mutating URLs in a timely manner [8]. To alleviate these drawbacks, subsequent studies attempted to combine domain blacklists with website behavioral analysis, introducing features such as registration information, IP addresses, and access frequency to improve detection accuracy, especially for known phishing pages [9]. Some studies also built classifiers based on heuristic rules and machine learning, using interpretable URL features such as IP address inclusion, port specification, hexadecimal usage, and special character ratio [10]. In addition, feature selection methods (e.g., information gain, recursive feature elimination with random forests or neural networks) have been adopted to enhance model stability [11]. For example, Zouina et al. proposed a lightweight phishing detection method based on SVM, with URL similarity metrics for discrimination, featuring simple implementation but heavy reliance on handcrafted features [12]. Vanitha et al. used logistic regression (LR) to model URL features for efficient and interpretable malicious URL detection [13]. Subasi presented a random forest (RF)-based detection model, which uses ensemble learning to strengthen decision tree stability and achieves consistent performance across experiments [14].

Nevertheless, such methods rely heavily on manual feature engineering (e.g., URL length, symbol ratio, and keywords), which limits feature dimension and expressiveness. They struggle to cope with diverse attack patterns and frequently evolving malware variants. With the advancement of deep learning, end-to-end representation learning has gradually become a major research direction for malicious URL detection. A large number of comparative studies have verified that deep learning models significantly outperform traditional machine learning methods in network security detection tasks by automatically learning hierarchical features without manual design. Convolutional Neural Networks (CNNs), in particular, have demonstrated superior accuracy and robustness in complex network environments compared with traditional machine learning models in network intrusion detection and malicious traffic identification, as validated in a systematic comparative study [15]. Gupta et al. combined RNN with the whale optimization algorithm for phishing URL detection, improving parameter tuning efficiency but yielding relatively low overall accuracy [16]. Xiao et al. proposed CNN-MHSA, which integrates convolution and multi-head self-attention (MHSA) and adaptively weights sequence segments to mine comprehensive URL features [17]. Le used URLNet to leverage convolutional neural networks to learn character-level and word-level URL representations, overcoming the over-reliance on manual features in traditional methods [18]. Kamran et al. presented CNN-GAN, which uses CNN for feature extraction within a generative adversarial framework and employs conditional GAN to assist simultaneous phishing URL generation and detection [19]. Liang et al. adopted BiLSTM to model dependencies in URL sequences via bidirectional recurrent structures, enhancing representation for complex URLs [20]. Maneriker et al. proposed URLTran, which uses Transformer to model subword-level sequences and performs binary classification based on the token, effectively reducing the false positive rate while maintaining a high recall rate [21].

Furthermore, TransURL employs a multi-layer Transformer encoding architecture and multi-scale semantic alignment to capture long-range character dependencies, thereby enhancing detection robustness against semantically obfuscated malicious URLs [22].

PMANet introduces a progressive multi-scale attention mechanism to construct hierarchical feature perception from the character level, word level, to structural level, strengthening the extraction of both local subtle patterns and global semantic correlations of URLs, thus improving detection performance on mutated and obfuscated samples [23].

In addition to sequence-based models, graph learning methods have also been applied to malicious URL detection. URL2Path proposes a robust graph learning framework that models URL path structures and semantic relationships as graph data, effectively capturing structural features of URLs and improving detection performance for variant samples [24].

Although the above methods have improved malicious URL detection performance to a certain extent, they still suffer from inherent limitations. On the one hand, traditional CNNs are restricted to local segment pattern extraction and struggle to capture long-range dependencies. Although Transformer-based attention models excel at global relationship modeling, they lack sufficient attention to local details and structural features, often leading to information redundancy. On the other hand, URLs usually consist of multi-level features including domain names, paths, and parameters. Most existing methods rely on a single feature level or static fusion strategies, lacking dynamic interaction modeling for multi-granularity information and failing to balance global dependency and local discriminative capability. In complex and ever-changing network environments, these drawbacks lead to insufficient feature extraction, reduced detection accuracy, and increased false positives and false negatives.

To address the above issues, this paper proposes PLG-URLNet, a progressive local–global interactive network for malicious URL detection. The core contribution of PLG-URLNet lies in the systematic fusion of convolutional neural networks and Transformers, together with a layer-wise interactive mechanism to achieve collaborative modeling of local patterns and global dependencies. Compared with existing works, the innovations of this paper are summarized as follows:

(1) Two-branch parallel architecture: We construct parallel convolutional and Transformer branches to extract local character patterns and global semantic dependencies of URLs, respectively. Different from the serial combination in CNN-MHSA [17], the dual-branch parallel design preserves feature independence during extraction and avoids information confusion.
(2) Progressive Local-Global Interaction Module (PLGIM): To overcome the limitation of single high-level fusion in existing methods, we design a bidirectional interactive mechanism embedded in each layer, enabling layer-wise alignment and co-evolution of local and global features. Through cross-branch attention and gating adjustment, PLGIM achieves complementary enhancement while retaining the discriminative information of each branch.
(3) Decoder-style fusion layer: Inspired by the Transformer decoder, we design a multi-level attention fusion module to deeply integrate layer-wise interactive features for joint discrimination of local details and global semantics.

The core value of the above design is that we apply a progressive, bidirectional, gated layer-wise interaction mechanism to malicious URL detection, which effectively alleviates the problem of separation between local and global information in existing methods. Through multi-level and progressive feature fusion, the problem of local and global information separation in the existing methods when dealing with confusing characters and structural variants is effectively alleviated.

In order to more clearly show the differences between the proposed method and the existing representative methods, Table 1 compares the proposed method from the dimensions of network structure, feature fusion method and interaction granularity.

**Table 1.** Technical comparison between the proposed method and the existing representative methods.

| Methods | Network Structure | Feature Fusion Method | Interaction | Key Features |
|---|---|---|---|---|
| URLTran-MHSA [17] | CNN + MHSA | Access attention after convolution | Single-layer fusion | Attention reweights convolutional features |
| URLTran [21] | Transformer | Self-attention | No fusion | Classification based on [CLS] |
| BiLSTM [20] | Bidirectional LSTM | No | No | Sequential forward and backward |
| PLG-URLNet | CNN + Transformer | Layer-by-layer interaction | Layer-by-layer bidirectional | Local and global coevolution |

It is worth noting that the malicious URL detection field commonly uses a random train–test split protocol, which may cause variant samples from the same phishing toolkit to appear in both training and test sets, thus overestimating the model's true generalization ability. In addition to using the same evaluation protocol as existing works for fair comparison, we further conduct time-sequential partitioning verification. Experimental results show that PLG-URLNet maintains a performance advantage in the stricter temporal generalization scenario.

## 2. PLG-URLNet Model Construction

In this paper, we propose a Progressive Local–Global Interaction Network for URL analysis (PLG-URLNet), which is built upon layer-wise interactions between a local convolutional branch and a global attention-based Transformer branch. The overall framework is illustrated in Figure 1. The model first converts the input URL into a character-level representation, which is subsequently mapped into a continuous embedding space through a convolution-based embedding layer. The resulting representation is then fed in parallel into both the Transformer encoder branch and the convolutional branch.

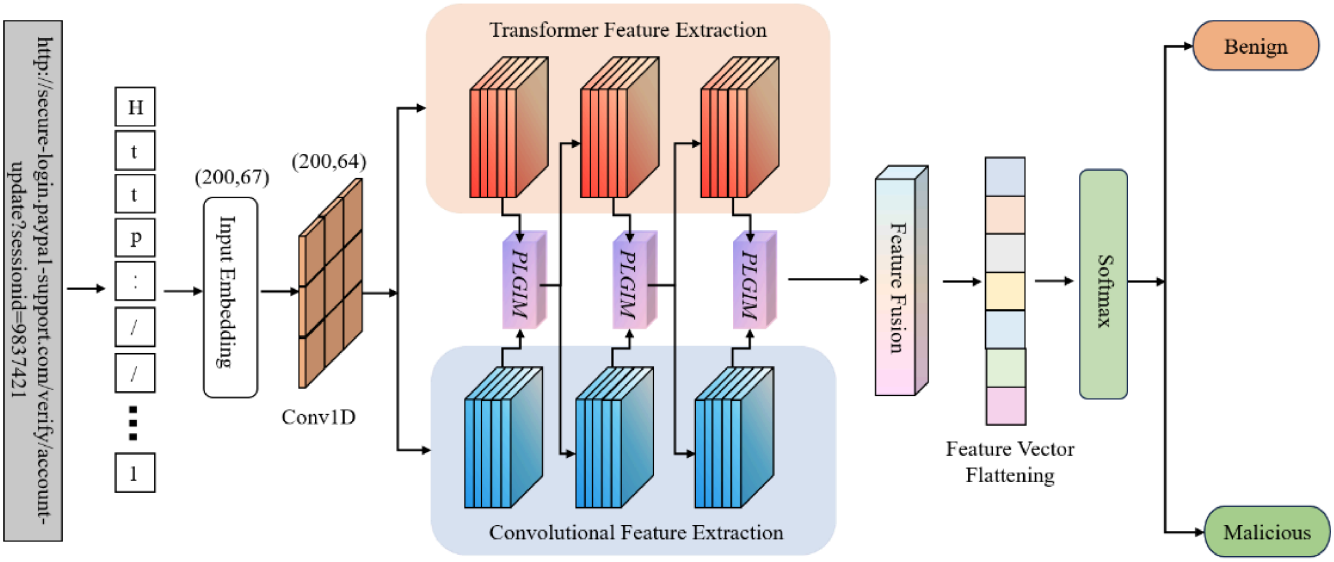



**Figure 1.** Architecture of the PLG-URLNet Model.

To enable dynamic alignment and effective feature fusion between local patterns and global dependencies across different depths, we design a Progressive Local–Global Interaction Module (PLGIM). This module performs bidirectional interaction between convolutional and attention-based features at each layer, enabling joint modeling of both representations. In this way, local and global features are progressively refined in a mutually reinforced manner throughout the network, resulting in a more robust and discriminative joint representation.

From a design perspective, the convolutional branch is responsible for capturing local patterns and short-range contextual information from character segments, whereas the Transformer branch models global dependencies and long-range contextual relationships. Through continuous interaction via the PLGIM at each layer, local features extracted by the convolutional branch provide precise boundary cues and structural constraints for the attention mechanism. Conversely, global dependencies derived from the Transformer branch guide the convolutional features to emphasize informative segments, thereby mitigating the limitations of single-path feature extraction and enabling effective integration of local and global information.

As shown in Figure 1, the overall PLG-URLNet pipeline operates as follows: the raw URL is first tokenized at the character level and transformed into a matrix-form embedding. Subsequently, the convolutional and Transformer branches operate in parallel to extract local and global features, respectively. Deep feature fusion is achieved through the layer-wise interaction module. Finally, the fused representation is passed to a lightweight decoder-style fusion layer and a classifier to produce the final detection results.

### *2.1. Data Preprocessing and Input Construction*

At the input stage of the model, we first convert the raw URLs into a sequence representation. We build an initial character dictionary with 67 simple characters, which includes 26 lowercase English letters, 10 digits, 30 most frequent special symbols, and 1 placeholder token. Afterward, all URLs are truncated or padded to be L = 200. Specifically, if a URL is longer than 200 characters, we keep only the first 200 characters; otherwise, we pad more placeholder symbols at the end of the URL. Therefore, we can get the input with the same dimension and enable the model to effectively handle the variable-length URLs in one batch.

Note that in practice, real data may include a significant number of infrequent or anomalous symbols, so the final dictionary is further extended by replacing infrequent characters with a single token [OOV] (out of vocabulary) and by adding [FILL] for padding the sequence. Finally, we obtain an effective alphabet of around m = 96 symbols, i.e., normalization and padding procedures result in lowering the noise from sparse features and improve the generalization properties of the model on datasets of different distributions.

To further improve representational capacity, an embedding matrix $E_m \in \mathbb{R}^{m \times k}$ is employed to map discrete characters into a dense vector space, where the embedding dimension is set to k = 32. Through this operation, each URL can be represented as

$$u_x \in R^{L \times d}, L = 200, \quad k = 32 \tag{1}$$

This dense embedding not only reduces the dimensionality of the original one-hot representation and thereby lowers subsequent computational overhead, but also enables the model to automatically learn latent semantic relationships between characters during training. As a result, the model's capability to distinguish between visually or structurally similar characters and variant URLs is significantly enhanced.

Building upon this representation, local contextual features are further extracted using a one-dimensional convolutional layer. The convolutional kernel size is set to 3 with a stride of 1, and the number of channels is projected from 32 to 64, resulting in a feature representation of size $200 \times 64$. The convolution operation not only strengthens feature expressiveness but also provides richer local pattern information to the subsequent Transformer branch through multi-channel feature mapping, which facilitates more effective modeling of global dependencies.

Truncation strategy analysis: The semantic information within URLs exhibits an uneven distribution, primarily concentrated at both ends. Specifically, the domain name

segment (i.e., the prefix) encodes site identity information, while the query parameter segment (i.e., the suffix) often contains malicious payloads or injected code. A naive truncation strategy that retains only the first L characters may result in the loss of critical information located in the tail segment. To address this issue, this study conducts a comparative analysis of different truncation strategies, evaluating three representative schemes:

- Scheme A: Pre-truncation (keep the first 200 characters);
- Scheme B: Post-truncation (keep the last 200 characters);
- Scheme C: Back and Forth truncation (keep first 100 characters + last 100 characters). The experiment was conducted on the validation set and the results are shown in Table 2.

**Table 2.** Comparison of experimental results of different truncation strategies.

| Truncation Strategies | Accuracy (%) | Precision (%) | Recall (%) | F1-Score (%) |
|---|---|---|---|---|
| Front truncation (Scheme A) | 98.59 | 98.80 | 98.36 | 98.58 |
| Posterior truncation (Option B) | 97.85 | 98.12 | 97.43 | 97.77 |
| Front and back truncation (Scheme C) | 98.43 | 98.67 | 98.12 | 98.39 |

The experimental results indicate that the head truncation strategy consistently outperforms both tail truncation and head–tail truncation across all evaluation metrics. This can be attributed to the fact that discriminative URL information—such as malicious keywords and suspicious domain names—is predominantly concentrated in the domain and path components, which are typically located in the head portion of the URL. Although the tail segment (i.e., query parameters) may also contain useful signals, its effectiveness is constrained by the fixed truncation length. In practice, retaining both head and tail information simultaneously may still lead to partial loss of critical contextual cues from each segment due to limited sequence length allocation. Consequently, the forward (head-based) truncation strategy is adopted in this study as the default preprocessing scheme.

### *2.2. Transformer Feature Extraction*

The Transformer branch is designed to model global dependencies within the URL character sequence, thereby compensating for the limitations of convolutional networks in long-range modeling. The core component of the Transformer branch is the encoder module, as illustrated in Figure 2. Transformers have achieved great success in many artificial intelligence fields, such as natural language processing, computer vision, and audio processing. Encoder module consists of a Multi-Head Self-Attention mechanism and a Feed-Forward Network (FFN), with two residual connections and layer normalization operations (Add and Norm) applied after each sub-layer [25,26]. During the input stage, the character embedding matrix $u_x \in R^{L\times d}$ is fed into the first encoder layer. The multi-head self-attention mechanism computes the correlations between different characters in the sequence. Specifically, for the input feature sequence $H^{(l-1)} \in R^{L\times d}$query (Q), key (K), and value (V) matrices are obtained via linear transformations. The scaled dot-product attention mechanism is then employed to calculate dependencies between different positions in the sequence.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (2)$$

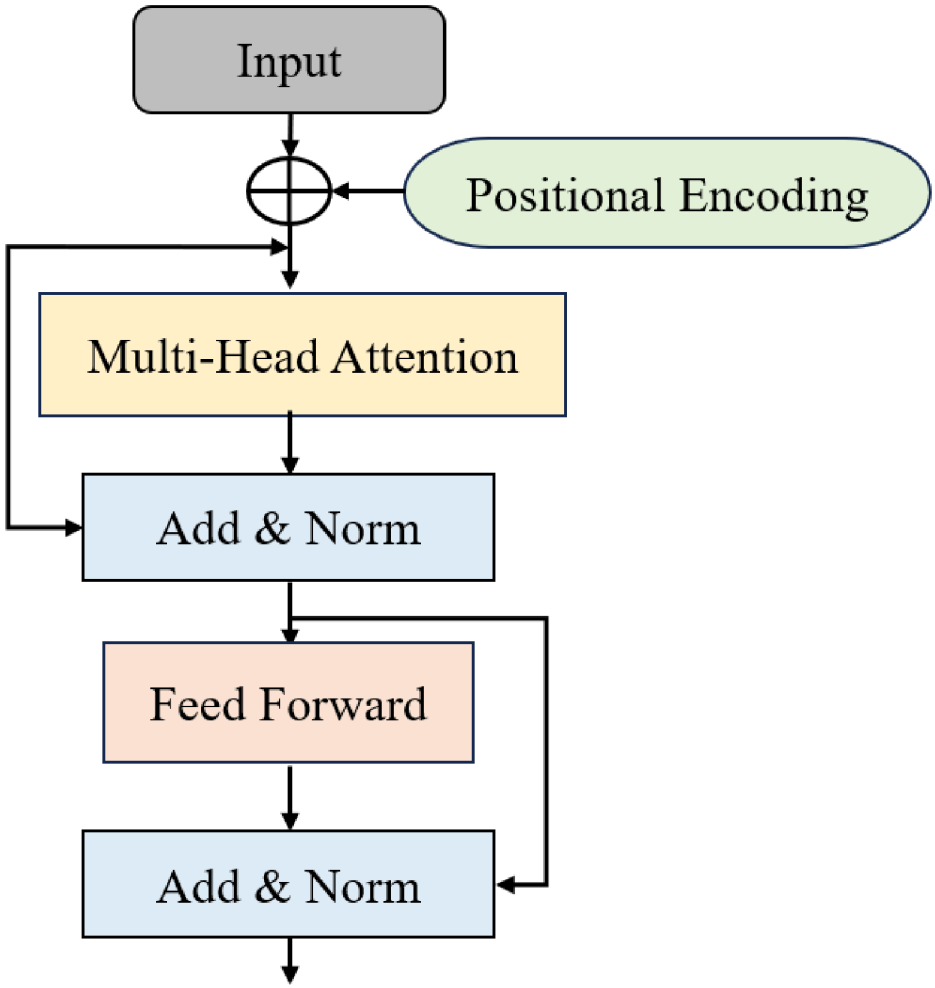



**Figure 2.** Transformer-based feature extraction module.

The multi-head attention mechanism performs multiple independent attention computations in parallel, concatenates their results, and projects them through an output transformation:

$$MHA() = Concat(head_1, \ldots, \ head_h)W^O \quad (3)$$

where h denotes the number of attention heads, and $W^O$ represents a trainable output projection matrix. This multi-head architecture enables the model to simultaneously learn diverse dependency patterns across distinct representation subspaces. To illustrate this capability, consider the URL example http://abc-school.net/login_portal: despite the significant positional distance between the subdomain component "abc-school" and the path segment "login_portal" within the character sequence, the self-attention mechanism effectively establishes their semantic relationship during the encoding process. This capacity for capturing long-range dependencies assists the model in identifying potential security threats that may be distributed across different segments of the URL structure, thereby enhancing the detection of semantically obfuscated malicious URLs through a comprehensive understanding of the global context.

After obtaining the output from the attention mechanism, the encoder introduces residual connections and layer normalization (Add and Norm) to enhance the stability of cross-layer information propagation and alleviate potential gradient vanishing or explosion in deep networks. The output is then passed through a Feed Forward Network (FFN),

which typically consists of two fully connected layers with a nonlinear activation function. The computation process is defined as follows:

$$FFN(X) = \sigma(XW_1 + b_1)W_2 + b_2 \tag{4}$$

where $\sigma(\cdot)$ denotes the ReLU activation function. The FFN serves to further perform nonlinear mapping and higher-level abstraction on the attention-processed features, thereby enhancing the model's ability to express complex feature relationships. The resulting features are again processed through residual connections and layer normalization, ensuring stable information flow across layers and yielding a more stable hierarchical representation.

Through hierarchical stacking of multiple encoder layers, the Transformer architecture progressively expands its contextual modeling capacity over URL sequences, enabling comprehensive representation of complex relational patterns across long character spans. In contrast to convolutional operations, which are primarily effective at extracting localized features, the Transformer mechanism is better suited for capturing domain-specific syntactic structures, hierarchical path organization, and semantic relationships within query parameters. This architectural distinction endows the model with complementary global representation capabilities that are essential for understanding sophisticated URL manipulation strategies, thereby providing a solid foundation for robust malicious URL detection through holistic sequence modeling.

### *2.3. CNN Feature Extraction*

The convolutional branch aims to capture local patterns and short-range dependencies within URL character sequences. Its core structure is illustrated in Figure 3, which consists of a one-dimensional convolutional layer (Conv1D-64), followed by normalization, a nonlinear activation function (ReLU), and a Dropout layer [27–29]. First, the input feature matrix undergoes one-dimensional convolution, where the kernel size is set to 3 and the number of channels to 64, enabling the model to extract local information from every three adjacent characters. The convolution operation can be formulated as

$$Z^{(l)} = Conv1D\left(U^{(l-1)}; W^{(l)}\right) + b^{(l)} \tag{5}$$

where $X^{(l-1)}$ represents the input feature matrix of the $(l-1)$-th layer, $W^{(l)}$ denotes the convolution kernel, and $b^{(l)}$ is the bias term. Through this operation, the convolutional branch learns local contextual representations from contiguous character fragments.

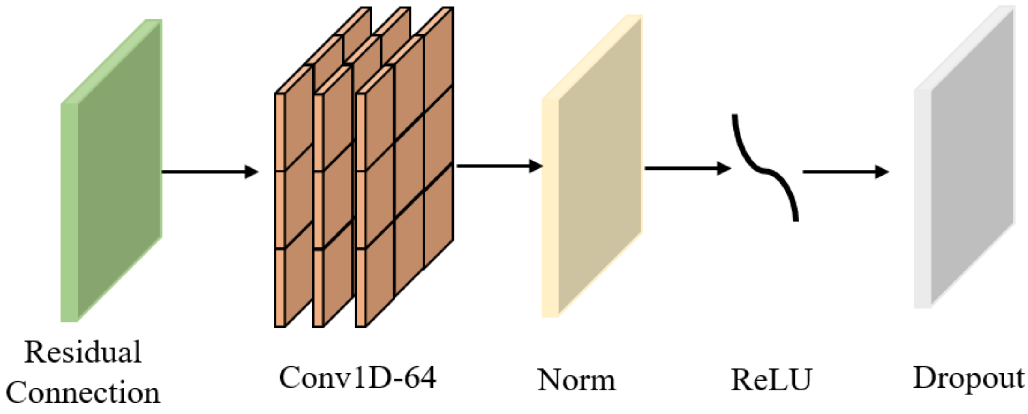



**Figure 3.** CNN-based feature extraction module.

The convolutional output is then passed through normalization and the ReLU activation function, defined as

$$A^{(l)} = ReLU\left(Norm\left(Z^{(l)}\right)\right) \tag{6}$$

Subsequently, a Dropout layer is applied:

$$H_c^{(l)} = Dropout\left(A^{(l)}\right) \tag{7}$$

By randomly deactivating a proportion of neurons during training, Dropout effectively alleviates overfitting and enhances the model's generalization capability.

Through the hierarchical stacking of multiple convolutional modules, the convolutional branch progressively expands its receptive field, enabling the capture of increasingly complex character-level patterns and contextual dependencies. In the initial layers, the network primarily learns elementary character n-grams such as "log", whereas deeper layers progressively integrate these local patterns into more semantically meaningful lexical units, for example "login". This hierarchical abstraction further enables higher layers to identify characteristic malicious fragments—such as "verify_account" and "secure-update"—which are frequently associated with phishing activities and malicious URL structures. Such a hierarchical design facilitates multi-level feature composition, allowing the model to effectively capture and discriminate potentially malicious semantic patterns in a structured and progressive manner.

### *2.4. Technology Inheritance and Positioning in This Paper*

The above review shows that malicious URL detection methods have experienced the evolution from manual feature engineering to end-to-end deep learning. Among the deep learning methods, CNN focuses on local pattern extraction [18], RNN/BiLSTM focuses on [20] sequence dependence, and Transformer is good at long-distance modeling [21]. In recent years, attempts to fuse multi-type network structures have made certain progress. For example, CNN-MHSA [17] combines convolution and attention serially, and URLTran [21] uses pure Transformer architecture.

Based on the above research, PLG-URLNet proposed in this paper systematically integrates and extends:

Inheritance: The design of the convolutional branch borrows the idea of character-level feature extraction from URLNet [18]. The Transformer branch refers to the self-attention mechanism of URLTran [21].

Improvement points: Different from the serial fusion mode of CNN-MHSA, this paper adopts a dual-branch parallel architecture to maintain the independence of local and global features in the extraction process. The Layer-by-layer interaction module (PLGIM) is introduced to realize cross-layer and bidirectional feature alignment, which makes up for the lack of representation ability of single-layer fusion. The fusion layer is designed based on the idea of Transformer decoder to realize deep feature integration.

Innovation positioning: The core contribution of this paper is not to propose a new network module, but to rationally combine and extend the existing technologies. Different from existing methods that use single-layer fusion, serial fusion, or simple concatenation, the proposed PLGIM performs progressive and bidirectional feature interaction at each network layer, which enables better cooperation between local convolution and global attention. The effectiveness of this design is verified through systematic experiments on obfuscated and variant URLs.

### *2.5. Progressive Local-Global Interaction Module*

To enable comprehensive integration of localized convolutional features and globally contextualized attention representations, this work introduces the Progressive Local-Global Interaction Module (PLGIM). Departing from conventional approaches that rely on isolated processing or single-stage fusion at predetermined network depths, PLGIM establishes a continuous two-way exchange mechanism between convolutional operations and Transformer encoders at every architectural layer. This interconnected architecture ensures that both feature types undergo reciprocal refinement and co-evolution throughout the network hierarchy, substantially strengthening the model's representational power. The complete module design and information flow are detailed in Figure 4.

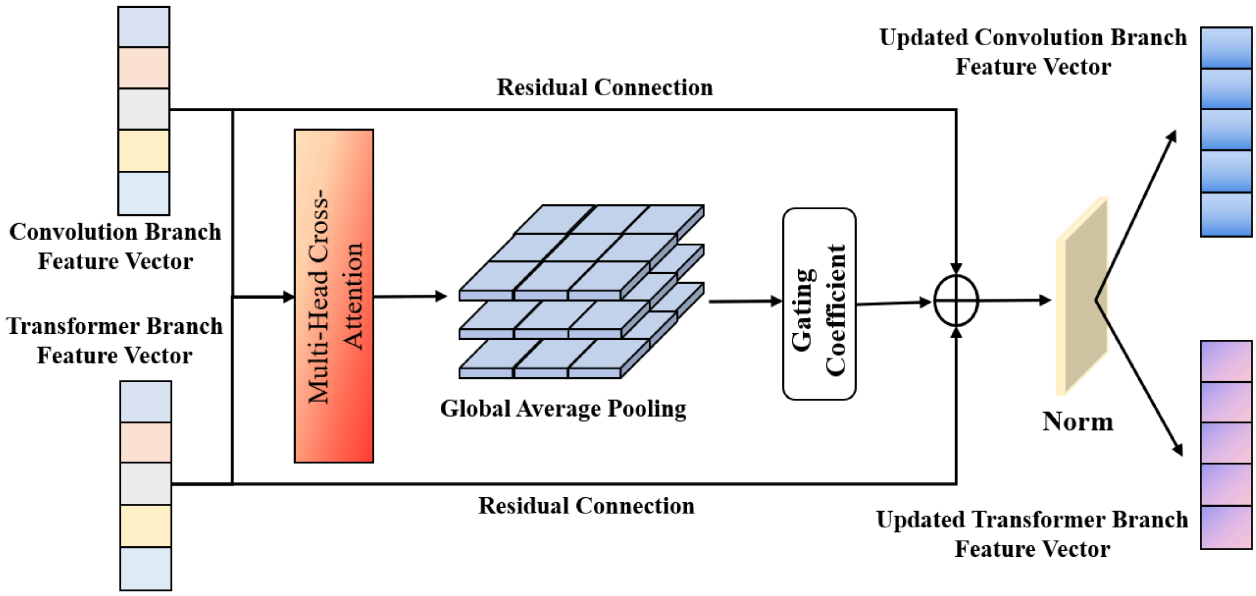



**Figure 4.** Progressive Local-Global Interaction Module (PLGIM).

At the $l$-th layer, let the outputs of the convolutional and Transformer branches be $F_c^{(l)}$ and $F_t^{(l)}$, respectively. PLGIM establishes bidirectional cross-branch attention between them as follows:

$$A_{c \leftarrow t} = softmax\left(\frac{H_c^{(l)} W_c^Q \left(H_t^{(l)} W_t^K\right)^T}{\sqrt{d}}\right) H_t^{(l)} W_t^V \tag{8}$$

$$A_{t \leftarrow c} = softmax\left(\frac{H_t^{(l)} W_t^Q \left(H_c^{(l)} W_c^K\right)^T}{\sqrt{d}}\right) H_c^{(l)} W_c^V \tag{9}$$

It is worth noting that the proposed interaction design fundamentally differs from conventional cross-modal attention mechanisms. In the URL detection scenario, the convolutional branch captures local patterns that naturally align with structural boundaries in URLs (e.g., "." and "/" separators), whereas the Transformer branch, with its global receptive field, is more effective at correlating suspicious keywords dispersed across path and query segments. The bidirectional interaction mechanism allows global contextual information derived from the convolutional branch to guide the Transformer in attending to non-adjacent malicious fragments, while local boundary cues provided by the convolutional branch help suppress the Transformer's tendency to over-attend to high-frequency but semantically uninformative tokens such as URL separators. This specialized coopera-

tion is a key factor underlying the superior performance of PLGIM compared with generic fusion strategies in obfuscated character recognition tasks.

In the interaction process between local and global features, naive full fusion without selective control may introduce redundancy and noise, leading to mutual interference between the two feature types. For instance, the Transformer branch may assign disproportionately high attention weights to irrelevant or weakly informative characters, such as frequent URL delimiters, which, if directly propagated to the convolutional branch, can degrade the discriminative power of local representations. To address this issue, a gating mechanism is introduced into PLGIM. Specifically, global semantic representations from both branches are aggregated via Global Average Pooling (GAP), and gating coefficients are computed to adaptively regulate the proportion of interactive information exchanged between the two branches.

The reason why GAP is selected over other pooling methods is that it can preserve the overall response strength of the feature map and is robust to the position shift of character-level features. At the same time, GAP is simple to calculate and does not introduce additional learnable parameters, which is suitable for deployment in resource-constrained scenarios. The gating coefficient is calculated as shown in Equations (10) and (11):

$$g_c = \sigma\left(W_g^c\left[GAP\left(H_c^{(l)}\right)\middle|\middle|GAP\left(H_t^l\right)\right] + b_g^c\right) \tag{10}$$

$$g_t = \sigma\left(W_g^t\left[GAP\left(H_t^{(l)}\right)\middle|\middle|GAP\left(H_c^l\right)\right] + b_g^t\right) \tag{11}$$

where $\sigma(\cdot)$ denotes the Sigmoid function, and $||$ represents vector concatenation. After obtaining the gating coefficients, the updated feature representations are defined as

$$\widetilde{H}_c^{(L)} = LN\left(H_c^{(l)} + g_c \odot A_{c\leftarrow t}\right) \tag{12}$$

$$\widetilde{H}_t^{(L)} = LN\left(H_t^{(l)} + g_t \odot A_{t\leftarrow c}\right) \tag{13}$$

where $\odot$ indicates element-wise multiplication, and LN refers to Layer Normalization. Through this residual-enhanced structure, PLGIM introduces cross-branch interaction features while preserving the integrity of the original information.

The proposed PLGIM integrates layer-wise bidirectional interaction, gating-based regulation, and residual normalization in a unified framework, thereby establishing a stronger coupling between local and global feature representations. This design effectively addresses the limitations of single-stage fusion methods, which typically suffer from insufficient representational capacity. Through a progressive inter-layer interaction mechanism, the module continuously reinforces the complementary strengths of convolutional and attention-based representations, enabling coordinated improvements in both the depth and breadth of feature modeling. As a result, PLGIM significantly enhances the model's ability to capture latent dependency structures within malicious URLs.

### *2.6. Fusing Layers with Classifiers*

Upon completion of progressive interaction and feature extraction, the convolutional and Transformer branches produce local and globally contextualized representations, respectively. A naïve fusion strategy, such as simple concatenation or element-wise weighted summation, would be insufficient to capture the complex interdependencies between local patterns and global contextual information, thereby leading to suboptimal feature integration. To address this limitation, we draw inspiration from the Transformer decoder architecture and design an enhanced fusion layer equipped with multi-level attention mechanisms. This architectural choice enables fine-grained feature alignment and context-aware

integration across heterogeneous representations, thereby improving both representational coherence and discriminative capability. This design is particularly aligned with the requirements of long-sequence modeling tasks, which demand strong predictive capacity and robust dependency modeling [30]. Overall, the proposed fusion strategy enhances the model's generalization ability while preserving its discriminative power. The complete structure of the fusion layer is illustrated in Figure 5.

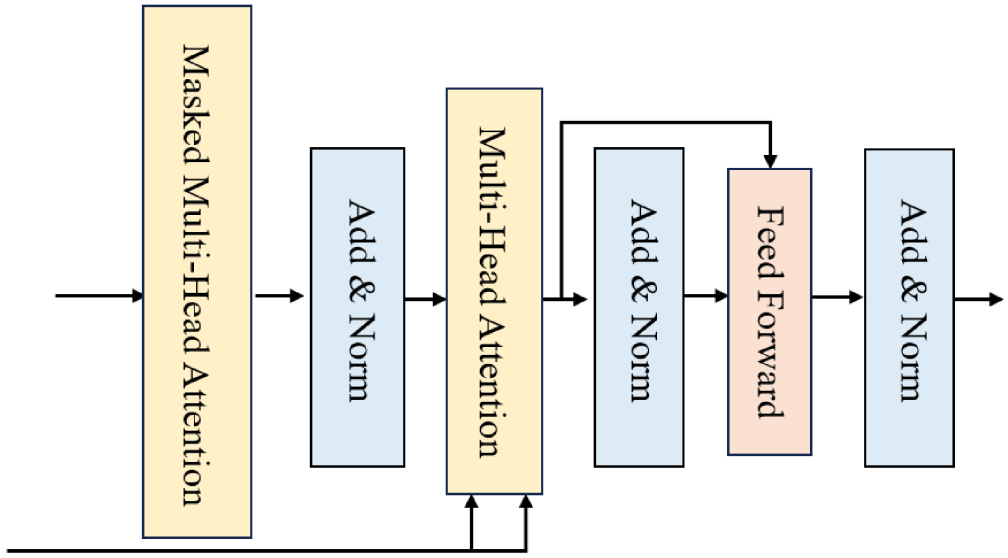



**Figure 5.** Fusion Layer and Classifier.

First, the outputs of the two branches are concatenated to form a unified representation, as shown in Equation (14):

$$H_{fusion} = [\widetilde{H}_c^{(L)} \Big|\Big| \widetilde{H}_t^{(L)}] \in R^{L \times 2d} \tag{14}$$

This combined representation is then successively processed by masked multi-head attention, multi-head attention, and a feed-forward network, which together suppress local noise, model global dependencies, and enhance nonlinear expressiveness. Each submodule is followed by a residual connection and layer normalization (Add and Norm) to ensure stable gradient propagation and fusion consistency. The final high-dimensional fused feature representation is obtained as defined in Equation (15):

$$O = Flatten\left(H'_{fusion}\right) W_f + b_f \tag{15}$$

In the above equations, $W$ and $b$ represent the weight matrix and bias term of the fully connected layers, respectively. The Softmax function is then applied to compute the prediction probability distribution, as shown in Equation (16):

$$\hat{y} = Softmax(O) \tag{16}$$

where $\hat{y}$ denotes the predicted probability that a given input URL belongs to the malicious or benign class. During training, the model employs the standard cross-entropy loss function, defined as follows:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{2} y_{i,k} \log(\hat{y}_{i,k}) \tag{17}$$

where $N$ represents the batch size, and $y_{ik}$ denotes the ground-truth label of the $i$-th sample in class $k$.

In contrast to conventional fusion approaches limited to elementary operations such as concatenation or weighted averaging, the proposed fusion layer incorporates multi-level attention mechanisms combined with residual normalization during the decoding stage. This sophisticated integration facilitates profound interaction between localized and globally contextualized representations while maintaining computational efficiency. The architectural design not only achieves substantial improvement in detection accuracy for malicious URLs but also demonstrates enhanced generalization capacity across diverse threat patterns, establishing an optimal balance between model complexity and operational effectiveness.

## 3. Experiments and Analysis

### *3.1. Hardware Environment and Parameter Settings*

All experiments are implemented on a Windows 11 64-bit operating system with an Intel Core i7-10700 CPU, an NVIDIA RTX 3090 GPU (24 GB VRAM), and 32 GB RAM. The software environment includes CUDA 11.2, Python 3.9, and PyTorch 2.0. To ensure experimental reproducibility, a fixed random seed of 42 is adopted for all random operations, including network initialization, data shuffling, and dataset splitting.

The character embedding dimension k is set to 32 to balance model expression ability and computational efficiency. Sensitivity experiments on embedding dimensions are shown in Table 3.

**Table 3.** Sensitivity Analysis of Embedding Dimensions.

| Embedding Dimension (k) | Number of Parameters (M) | Accuracy (%) | F1-score (%) | Inference Time (ms/Sample) |
|---|---|---|---|---|
| 16 | 0.48 | 97.82 | 97.71 | 8.2 |
| 32 | 0.82 | 98.59 | 98.58 | 12.5 |
| 64 | 1.54 | 98.63 | 98.61 | 21.3 |

Experimental results show that when k increases from 16 to 32, the accuracy improves by 0.77 percentage points, while the increase from 32 to 64 brings only 0.04 percentage points improvement but nearly doubles the parameters and inference time. Therefore, k = 32 is chosen as the embedding dimension. The input URL sequence length is uniformly normalized to L = 200 characters. The model is optimized using the Adam optimizer with a learning rate of $1 \times 10^{-4}$ and a batch size of 32. The Transformer encoder consists of 2 layers with 4 attention heads and a hidden dimension of 128. A dropout rate of 0.1 is applied to mitigate overfitting. The model is trained for a maximum of 30 epochs, with early stopping (patience = 5) based on the validation loss. A fixed random seed of 42 is used to ensure reproducibility.

### *3.2. Dataset, Stratified Partitioning and Class Imbalance Processing*

To evaluate the effectiveness of the proposed method, experiments are conducted on three public malicious URL datasets: PhishTank, UCI PhiUSIIL, and Kaggle. The PhishTank and UCI PhiUSIIL datasets are merged as the training–validation source domain, and the Kaggle dataset is used as a completely independent out-of-distribution (OOD) test set. The merged source dataset is divided into training and validation sets at a fixed ratio of 8:2 using stratified random sampling based on the benign/malicious label, and a fixed random seed of 42 is adopted for all data shuffling and splitting to ensure reproducibility. After partitioning, the training set contains 38,736 URL samples, the validation set contains 9684

URL samples, and the independent OOD test set (Kaggle) contains 8000 URL samples. To assess temporal generalization, time-sequential partitioning is performed only on the Kaggle dataset: all URLs are sorted by public submission timestamp in ascending order, with the first 80% of early samples used for training and the last 20% of recent samples used for testing, and no data from PhishTank or UCI PhiUSIIL is involved in this protocol.

To ensure reliable evaluation, exact duplicates and near-duplicate URLs are removed before splitting by direct discarding and Jaccard similarity filtering (>0.95) on normalized character sequences, and deduplication is carried out independently on each dataset. The benign-to-malicious ratio is adjusted to approximately 1:1 in both training and validation sets, and a weighted cross-entropy loss is used to address class imbalance. All URLs are uniformly normalized to fixed-length sequences of L = 200 characters by pre-truncation or [FILL] padding. Strict data leakage prevention measures are implemented: the Kaggle test set is never used for training, validation or parameter tuning; the vocabulary, embedding matrix and normalization parameters are learned only from the training set; no sample sharing occurs between the source domain and the OOD test set; and future test data is completely invisible to the training process in the time-ordered split.

### *3.3. Evaluation Metrics*

In order to analyze the performance of detection as thoroughly as possible, we use the following four common metrics: accuracy, precision, recall, and F1-score. All of these metrics are rooted in the following four simple quantities: true positive (*TP*), true negative (*TN*), false positive (*FP*), and false negative (*FN*). These measures are sensitive to both the general overall accuracy and the ability of the model to detect malicious samples. Their formulas are shown below.

Accuracy is one of the most commonly used evaluation metrics in classification tasks. It measures the overall proportion of correctly predicted samples among all samples, defined as

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{18}$$

where *TP* denotes the number of true positives, *TN* the number of true negatives, *FP* the number of false positives, and *FN* the number of false negatives. Accuracy intuitively reflects the model's overall classification capability across all samples.

*Precision* measures the proportion of samples predicted as malicious that are actually malicious. It is defined as

$$Precision = \frac{TP}{TP + FP} \tag{19}$$

*Recall* indicates the proportion of all actual malicious samples that are successfully identified by the model. It is defined as

$$Recall = \frac{TP}{TP + FN} \tag{20}$$

The *F1-score* serves as a comprehensive metric combining precision and recall. It provides a balanced measurement when there is a trade-off between the two and is defined as the harmonic mean of precision and recall:

$$F1_score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{21}$$

### *3.4. Experimental Results and Comparative Analysis*

As shown in Figures 6 and 7, as the number of training iterations increases, the model error gradually decreases and converges, while the training accuracy steadily improves. These results indicate that the model is capable of effectively learning discriminative

features from the training data to distinguish between benign and malicious URLs, thereby enhancing its overall classification performance.

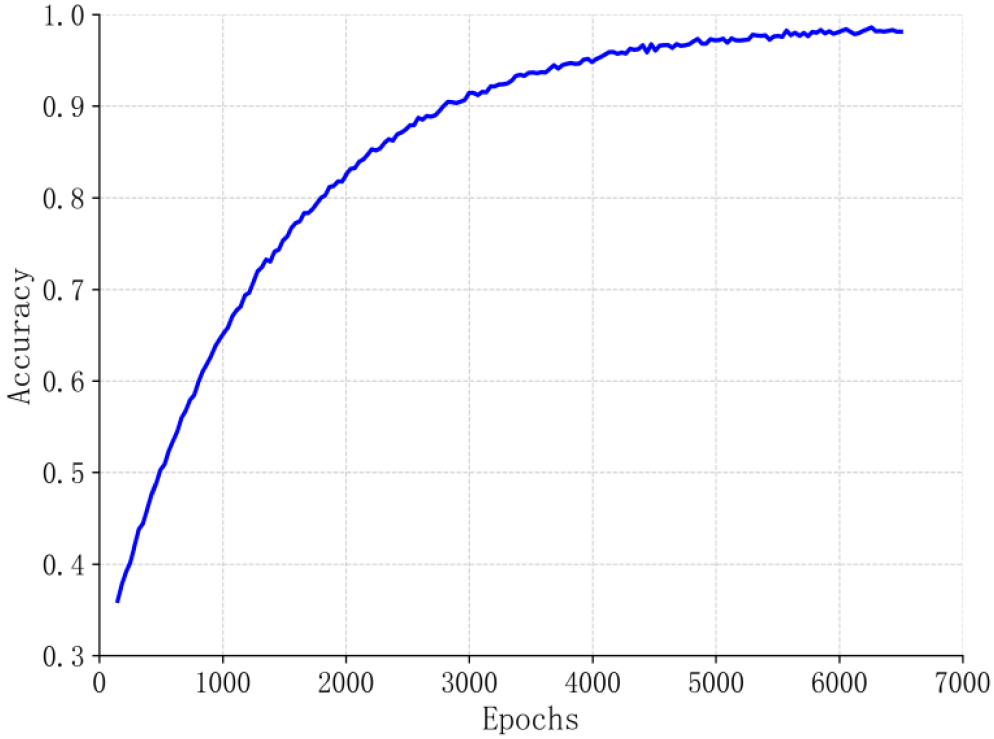



**Figure 6.** Training and validation accuracy curves of PLG-URLNet.

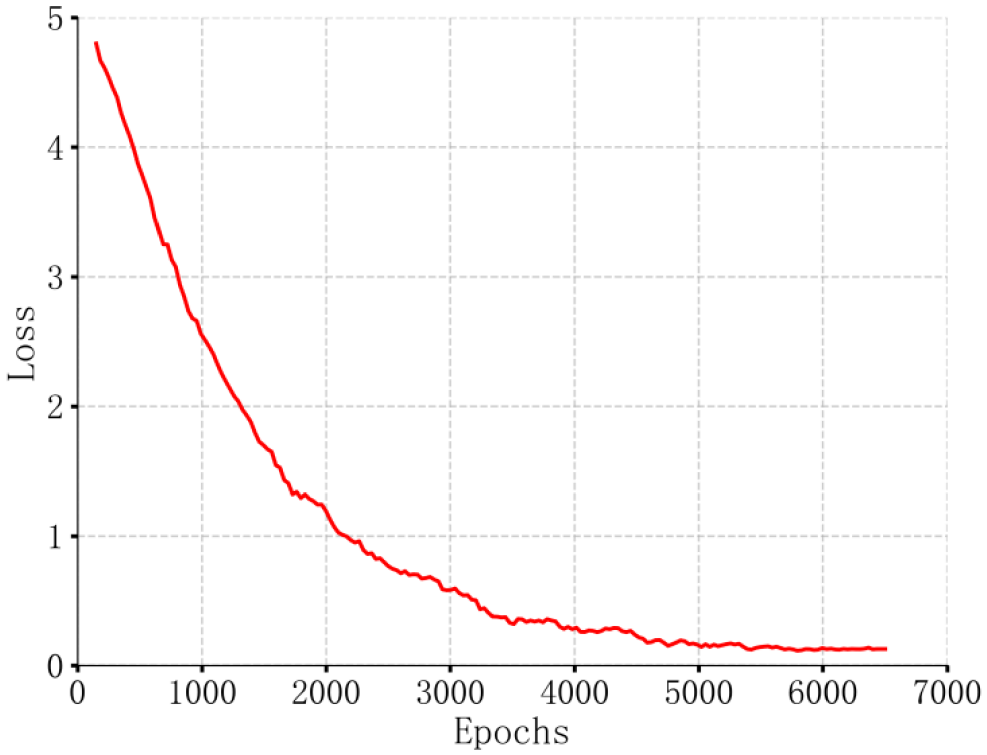



**Figure 7.** Training and validation loss curves of PLG-URLNet.

To ensure the reliability of the evaluation results and examine the model's temporal generalization ability, we first present a performance comparison under temporal partitioning before reporting the random partitioning results. As shown in Table 4, although

temporal partitioning leads to varying degrees of performance degradation for all methods, PLG-URLNet's relative advantage actually increases from 0.14% (vs. RoBERTa-URL) under random partitioning to 0.43%. This indicates that the layer-by-layer interaction mechanism helps capture temporally stable malicious semantic features. Random partitioning uses the PhishTank + UCI fusion data to randomly partition the training/validation set in an 8:2 stratified manner, with Kaggle serving as an independent test set; temporal partitioning uses the Kaggle dataset for training in the first 80% of the time and testing in the last 20%. Here, we present the performance of PLG-URLNet under the random partitioning protocol as a benchmark for comparison with temporal partitioning.

**Table 4.** Experimental Results Divided by Chronological Order.

| Division Methods | Accuracy (%) | Precision (%) | Recall (%) | F1-Score (%) |
|---|---|---|---|---|
| Random Division(Original) | 98.59 | 98.80 | 98.36 | 98.58 |
| Time-Order Division | 97.86 | 98.12 | 97.43 | 97.77 |
| Difference | −0.73 | −0.68 | −0.93 | −0.81 |

To comprehensively evaluate the effectiveness of the proposed PLG-URLNet model, we conduct a series of comparative experiments against a diverse set of baseline methods. These baselines include both traditional machine learning approaches and representative deep learning models commonly used in URL-based classification tasks. All baseline methods are reimplemented or adopted from their official implementations and evaluated under unified experimental conditions to ensure a fair comparison. Specifically, all models share the same data preprocessing pipeline, including URL truncation/padding, vocabulary construction, and embedding settings. In addition, training configurations such as batch size, optimizer type, learning rate, and hardware environment are kept as consistent as possible across all models. For parameter counts and inference time, values marked with "*" are estimated based on the original publications or standard model configurations, while the results for PLG-URLNet and directly reimplemented baselines are obtained through actual measurements under the same hardware and software environment. The selected baselines include traditional machine learning methods such as SVM [12], LR [13], and RF [14], as well as deep learning models including RNN [16], CNN-MHSA [17], BiLSTM [20], and URLTran [21]. To further assess the advancement of PLG-URLNet, we additionally include several recent state-of-the-art approaches, including BERT-URL [31], GraphURL [32], and RoBERTa-URL [33], which represent different modeling paradigms such as pre-trained language models and graph-based methods. The test results are shown in Table 5.

Table 5 shows that the proposed PLG-URLNet outperforms all compared methods in terms of accuracy, recall, F1 score, and precision, including methods based on pre-trained models (BERT-URL, RoBERTa-URL) and methods based on graph neural networks (GraphURL). Specifically, compared to RoBERTa-URL, PLG-URLNet improves accuracy by 0.14 percentage points and F1 score by 0.23 percentage points, indicating that under the premise of lightweight design, the layer-by-layer interaction mechanism in this paper can achieve detection performance superior to large pre-trained models.

Analysis reveals that the CNN-MHSA model outperforms pure sequence models in both accuracy and F1 score, indicating that convolutional structures have a certain advantage in capturing local anomalous character patterns in URLs. Meanwhile, URLTran demonstrates a more balanced performance between precision and recall, showcasing the effectiveness of self-attention-based models in modeling global URL dependencies. Through self-attention, the model can more fully learn long-distance semantic associations within URL sequences, thereby improving overall recognition capabilities.

**Table 5.** Performance comparison of different models.

| Model | Accuracy | Recall | F1 Score | Precision | Number of Parameters (M) | Inference Time (ms) |
|---|---|---|---|---|---|---|
| LR [13] | 95.57 | 94.91 | 95.51 | 96.12 | ~0.01 | N/A |
| SVM [12] | 96.21 | 95.72 | 96.10 | 96.48 | ~0.01 | N/A |
| RF [14] | 96.18 | 94.08 | 95.24 | 96.43 | ~0.01 | N/A |
| RNN [16] | 97.42 | 96.57 | 97.14 | 97.71 | ~1.2 | ~18.6 |
| BiLSTM [20] | 97.81 | 96.86 | 97.38 | 97.91 | ~2.4 | ~22.3 |
| CNN-MHSA [17] | 98.11 | 97.49 | 97.70 | 97.92 | ~1.8 | ~16.8 |
| URLTran [21] | 98.24 | 97.92 | 98.11 | 98.31 | ~3.6 | ~25.1 |
| BERT-URL [31] | 98.37 | 98.05 | 98.23 | 98.41 | ~110.0 | ~42.7 |
| GraphURL [32] | 98.29 | 97.98 | 98.14 | 98.30 | ~8.5 | ~28.4 |
| RoBERTa-URL [33] | 98.45 | 98.18 | 98.35 | 98.52 | ~82.5 | ~34.5 |
| PLG-URLNet | 98.59 | 98.36 | 98.58 | 98.80 | 0.82 | 12.5 |

Note: Parameter data marked with "~" are derived from the original publications or estimated based on standard model configurations. Inference time is measured under a unified hardware environment. "N/A" indicates that the inference time is not reported in the original work or not evaluated under the same experimental conditions.

While achieving optimal detection accuracy, PLG-URLNet uses only about 1% of the parameters of RoBERTa-URL, reducing inference latency by approximately 64%. This efficiency advantage makes the model easier to deploy in resource-constrained scenarios such as real-time traffic detection or edge computing.

Building on this, the proposed PLG-URLNet achieves collaborative modeling of local structural information and global semantic information of URL sequences by layer-by-layer interactive fusion of local convolutioThis further demonstrates the stability and reliability of PLG-URLNet [34] nal feature extraction and global attention mechanisms. Experimental results show that PLG-URLNet achieves the best performance across all evaluation metrics, with Accuracy and Precision reaching 98.59% and 98.80%, respectively. Compared with other comparative models, this model improves recall while maintaining high precision, thus gaining an advantage in the overall F1-score. This demonstrates a clear complementarity between local feature modeling and global semantic modeling [35]. By constructing a layer-by-layer local-global interaction mechanism, the potential discriminative information in URL sequences can be more fully mined, resulting in more stable and superior detection performance in malicious URL identification tasks.

To ensure the reliability of the experimental results, this paper conducted 10 repeated experiments on PLG-URLNet and the optimal comparison method (RoBERTa-URL), calculated the mean and standard deviation of each index, and used paired t-tests for statistical significance analysis. The experimental results are shown in Table 6.

**Table 6.** Results of statistical significance test.

| Model | Accuracy (%) | Standard Deviation | Recall (%) | Standard Deviation | F1 Score (%) | Standard Deviation |
|---|---|---|---|---|---|---|
| RoBERTa-URL | 98.45 | 0.12 | 98.18 | 0.15 | 98.35 | 0.14 |
| PLG-URLNet | 98.59 | 0.10 | 98.36 | 0.12 | 98.58 | 0.11 |
| t-value | 3.42 | - | 2.98 | - | 3.15 | - |
| *p*-value | 0.007 | - | 0.015 | - | 0.011 | - |

The t-test results show that the differences between PLG-URLNet and RoBERTa-URL on all metrics are statistically significant ($p < 0.05$), indicating that the performance improvement of our proposed method is not accidental and is statistically significant.

In addition to the classification metrics at fixed thresholds, this paper also uses ROC curves and their corresponding AUC values to evaluate model performance, as shown

in Table 7. The ROC curve, with the False Positive Rate (FPR) on the horizontal axis and the True Positive Rate (TPR) on the vertical axis, characterizes the model's classification performance across the entire threshold range by traversing all possible thresholds, thus reflecting the model's overall ranking ability and discriminative stability.

**Table 7.** Comparison of TPR@FPR and AUC at low false alarm rates.

| Methods | ROC-AUC | TPR@FPR = 0.01 | TPR@FPR = 0.001 |
|---|---|---|---|
| LR | 0.9228 | 0.7486 | 0.2421 |
| SVM | 0.9314 | 0.8189 | 0.2764 |
| RF | 0.9692 | 0.9387 | 0.4325 |
| RNN | 0.9418 | 0.8306 | 0.3097 |
| BiLSTM | 0.9739 | 0.9431 | 0.4916 |
| CNN-MHSA | 0.9761 | 0.9524 | 0.4688 |
| URLTran | 0.9687 | 0.8895 | 0.4129 |
| PLG-URLNet | 0.9856 | 0.9718 | 0.7034 |

Figure 8 shows the ROC curves of different models on the test set. Overall, PLG-URLNet's curve is higher than other models in most threshold ranges, and closer to the upper left corner, indicating that it maintains superior discrimination ability under different threshold settings. Its corresponding ROC-AUC value reaches 0.9856, significantly higher than traditional machine learning methods (such as LR 0.9228, SVM 0.9314) and deep learning baseline models [36] (RNN 0.9418, BiLSTM 0.9739, CNN-MHSA 0.9761, URLTran 0.9687). This demonstrates that the layer-by-layer local convolution and global attention interaction mechanism can effectively enhance the model's overall ranking ability for malicious URLs.

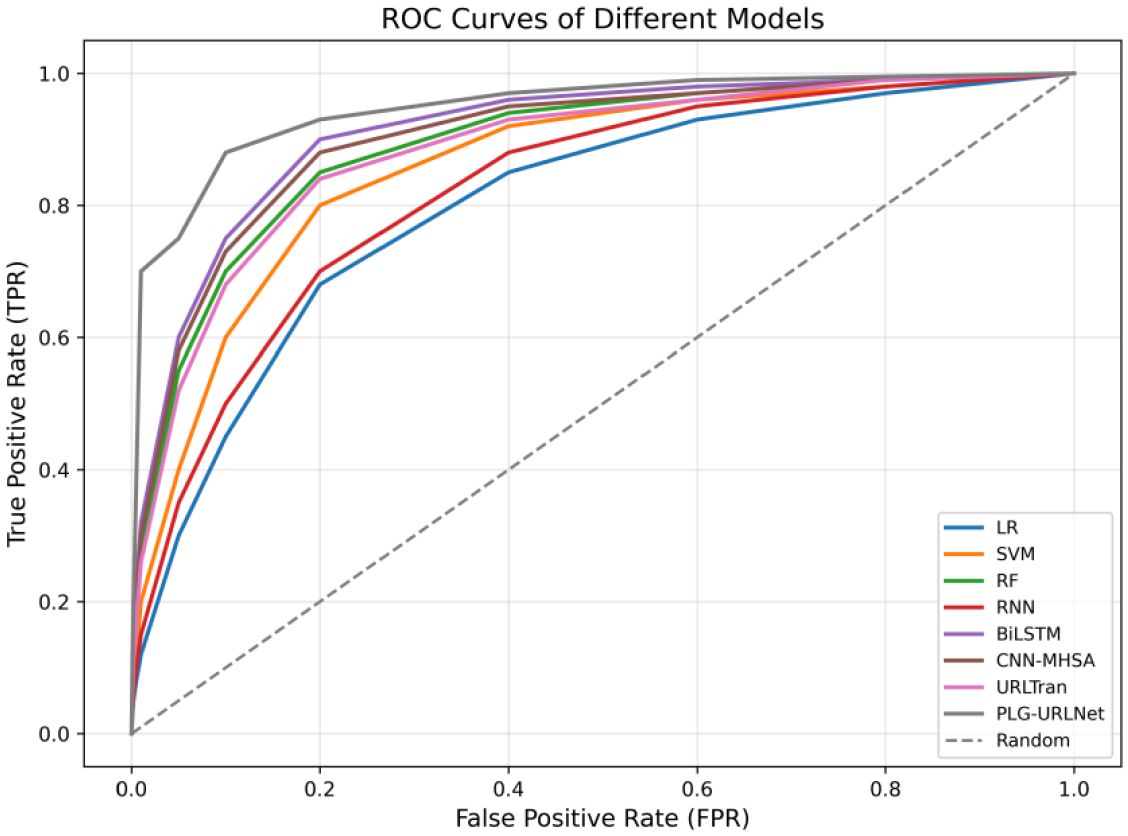



**Figure 8.** Comparison of ROC curves for different models.

Examining the detection performance at specific runtimes, PLG-URLNet achieved a TPR of 0.9718 at FPR = 0.01, higher than models such as CNN-MHSA (0.9524), BiLSTM (0.9431), and RF (0.9387). Even under the more stringent condition of FPR = 0.001, its TPR remained at 0.7034, significantly outperforming other methods (e.g., BiLSTM 0.4916, CNN-MHSA 0.4688, and RF 0.4325). This result demonstrates that even with a constrained false positive rate, PLG-URLNet can maintain a high malicious sample detection capability, exhibiting stronger robustness.

### *3.5. Ablation Study*

To verify the necessity of the gating mechanism in PLGIM, this paper designs a comparative experiment, comparing the complete PLGIM with the ungated version (i.e., directly fusing the two-branch features without gating coefficients). The experimental results are shown in Table 8.

**Table 8.** Experimental Results of Gated Mechanism Ablation.

| Model Variants | Accuracy (%) | Precision (%) | Recall (%) | F1-Score (%) |
|---|---|---|---|---|
| Full PLG-URLNet | 98.59 | 98.80 | 98.36 | 98.58 |
| Ungated PLG-URLNet | 98.21 | 98.45 | 97.92 | 98.18 |
| Difference | +0.38 | +0.35 | +0.44 | +0.40 |

Experimental results show that the model improves across all metrics by approximately 0.4 percentage points after introducing the gating mechanism, verifying its effectiveness in suppressing noise propagation and improving feature fusion quality.

To analyze the impact of different modules on the model's detection performance, this paper designed multiple ablation experiments, as shown in Table 9. The convolutional network, Transformer network, layer-by-layer feature interaction module, and fusion layer were validated using a controlled variable approach. The experimental results are shown in Table 10.

**Table 9.** Fine-grained ablation experiment design of PLG-URLNet modules.

| Number | Model Variants | CNN | Transformer | Simple Concatenation | PLGIM | Fusion Layer | Description |
|---|---|---|---|---|---|---|---|
| M1 | CNN only | √ | × | - | - | - | Benchmark 1 |
| M2 | Transformer only | × | √ | - | - | - | Benchmark 2 |
| M3 | CNN + Transformer | √ | √ | √ | × | × | Last layer only |
| M4 | CNN + Transformer + PLGIM | √ | √ | × | √ | × | Layer-by-layer interaction |
| M5 | CNN + Transformer + Fusion Layer | √ | √ | × | × | √ | Final fusion only |
| M6 | PLG-URLNet | √ | √ | × | √ | √ | Layer-by-layer interaction+ final fusion |

**Table 10.** Ablation experiment results of PLG-URLNet modules.

| Number | Model Variants | Accuracy (%) | Recall (%) | F1 Score (%) | Precision (%) |
|---|---|---|---|---|---|
| M1 | CNN only | 94.50 | 94.61 | 95.80 | 97.01 |
| M2 | Transformer only | 96.13 | 96.65 | 97.45 | 98.27 |
| M3 | CNN + Transformer | 97.28 | 96.89 | 97.15 | 97.42 |
| M4 | CNN + Transformer + PLGIM | 97.92 | 97.01 | 97.77 | 98.55 |
| M5 | CNN + Transformer + Fusion Layer | 97.85 | 97.12 | 97.69 | 98.31 |
| M6 | PLG-URLNet | 98.59 | 98.36 | 98.58 | 98.80 |

Comparing M1 and M2, the Transformer branch outperforms the CNN branch on its own, indicating that long-range dependency modeling is more crucial for URL detection.

Comparing M3 and M1/M2, simple feature concatenation and fusion outperforms a single branch, validating the complementarity of local and global features. Comparing M4 and M3, introducing PLGIM improves accuracy by 0.64 percentage points, validating the effectiveness of the layer-by-layer interaction mechanism. Comparing M5 and M3, introducing a fusion layer improves accuracy by 0.57 percentage points, validating the role of the decoding-based fusion layer. Comparing M6 and M4/M5, PLGIM and the fusion layer have synergistic gains, with optimal performance when used simultaneously. The ablation experiment shows that M4 significantly improves accuracy by 0.64% compared to M3, empirically demonstrating the superiority of this specialized design over simple feature concatenation.

In addition to the ablation study, to provide a more comprehensive understanding of the model performance, we further analyze the prediction outcomes at a finer granularity. Specifically, we examine the distribution of true positives (TPs), true negatives (TNs), false positives (FPs), and false negatives (FNs) on the test set. The results show that the proposed model maintains a low misclassification rate for both benign and malicious URLs, with balanced performance across classes. This further demonstrates the stability and reliability of PLG-URLNet.

### *3.6. Cross-Dataset Testing and Distribution Mismatch Analysis*

To further validate the generalization and robustness of PLG-URLNets in real-world scenarios, we conducted a strict cross-dataset test to evaluate its performance on out-of-distribution (OOD) samples and distribution mismatch. In practical network security applications, URL data collected from different platforms show significant differences in character distribution, domain structure, obfuscation strategies, and attack patterns, leading to obvious distribution differences between the training set and test set. Such domain shift often leads to performance degradation of deep learning models.

In this experiment, we adopt a strictly independent cross-dataset setting:

(1) Training domain: PhishTank + UCI PhiUSIIL (source dataset)
(2) Test domain: Kaggle (completely independent target dataset, not involved in training)

We compare PLG-URLNet with CNN-MHSA, URLTran, GraphURL, and RoBERTa-URL under OOD conditions. The results are shown in Table 11.

**Table 11.** Performance under cross-dataset (OOD) test.

| Model | Accuracy (%) | F1-Score (%) | ROC-AUC (%) | TPR@FPR = 0.001 |
|---|---|---|---|---|
| CNN-MHSA | 96.74 | 96.68 | 97.12 | 0.4126 |
| URLTran | 97.12 | 97.05 | 97.45 | 0.4783 |
| GraphURL | 97.29 | 97.23 | 97.76 | 0.5317 |
| RoBERTa-URL | 97.36 | 97.31 | 97.89 | 0.5641 |
| PLG-URLNet | 97.86 | 97.77 | 98.32 | 0.6852 |

Experimental results show that all models experience performance degradation under cross-dataset conditions due to distribution mismatch. However, PLG-URLNet still achieves the best performance in all metrics. Compared with RoBERTa-URL, PLG-URLNet improves accuracy by 0.5 percentage points and shows significant advantages in detecting unknown malicious samples under extremely low false alarm rates.

The strong OOD generalization ability mainly benefits from the Progressive Local-Global Interaction Module (PLGIM), which enables the model to capture domain-invariant malicious features rather than overfitting to dataset-specific characteristics. Through layer-by-layer fusion of local structure and global semantics, PLG-URLNet effectively alleviates

the performance drop caused by distribution shift and maintains stable detection for obfuscated and variant URLs in real open scenarios.

## 4. Conclusions

This paper addresses the challenges of fragmented local and global feature representation, as well as the difficulty in balancing dynamism and discriminability in malicious URL detection. To this end, we propose a layer-wise local–global interaction-based detection model, namely PLG-URLNet. The proposed method fully exploits the complementary strengths of convolutional and Transformer branches, enabling the simultaneous capture of local character-level patterns, short-range contextual features, global dependencies, and long-range contextual relationships. By introducing the Progressive Local-Global Interaction Module (PLGIM), the model achieves bidirectional alignment and joint modeling of local and global features across different network depths, thereby progressively enhancing feature representation capability. Experimental results demonstrate that the proposed model consistently outperforms comparative baselines on multi-source datasets, achieving a final detection accuracy of 98.59%. In addition, time-series partitioning experiments show that PLG-URLNet exhibits less performance degradation under concept drift conditions compared with existing methods, providing preliminary evidence of its robustness and adaptability to rapidly evolving threats in real-world network environments. Overall, PLG-URLNet demonstrates superior performance in jointly modeling local patterns and global dependencies, effectively mitigating the challenges posed by obfuscated characters and URL variants, while maintaining strong adaptability in complex network scenarios.

Future work will focus on two main directions. First, we will explore the integration of knowledge graphs with contextual behavioral features to further enhance the model's semantic understanding in complex attack scenarios. Second, we plan to incorporate adversarial sample generation and detection mechanisms to improve model robustness against advanced evasion attacks.

**Author Contributions:** Conceptualization, Y.X.; methodology, Y.X. and Y.L.; software, Y.X.; validation, Y.L. and Y.D.; formal analysis, Y.L. and Y.D.; investigation, H.L. (Hongzhi Liu)and H.L. (Heng Liu) ; resources, L.H.; data curation, L.H. and H.L. (Heng Liu); writing-original draft preparation, Y.X.; writing-review and editing, Y.H. and S.P.; visualization, H.L. (Hongzhi Liu) and S.P.; supervision, Y.H. and L.H.; project administration, L.H.; funding acquisition, Y.H. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the "Sensitive Data Discovery and Fine-Grained Access Control Technology" Project (Grant No. 2024MU072), funded by the Higher Education Research & Development Center, Ministry of Education, China.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data used in this study are publicly available benchmark datasets for URL detection.

**Conflicts of Interest:** Author Yongcai Xiao, Heng Liu, YuXin Deng, Shuhui Pan and HongZhi Liu were employed by the company State Grid Jiangxi Electric Power Research Institute. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

## References

1. Wu, S.; Luo, X.; Wang, W.; Qin, Y. Malicious URL detection method integrating multiple features. *J. Softw.* **2021**, *32*, 2916–2934. (In Chinese)
2. APWG. Phishing Activity Trends Report: 4th Quarter 2024. APWG: Lexington, MA, USA, 2025.

3. Wang, M.; Zang, X.; Cao, J.; Zhang, B.; Li, S. PhishHunter: Detecting camouflaged IDN-based phishing attacks via Siamese neural network. *Comput. Secur.* **2024**, *138*, 103668. [CrossRef]
4. Zhang, Y.; Xu, C.; Shi, F.; Hu, M.; Zhang, M.; Li, Y.; Xie, Z. Understanding and characterizing the adoption of internationalized domain names in practice. *IEEE Trans. Dependable Secur. Comput.* **2024**, *22*, 34–48. [CrossRef]
5. Fu, X.; Li, H.; Zhao, X. A survey on phishing recognition research. *Chin. J. Netw. Inf. Secur.* **2020**, *6*, 1–10. (In Chinese)
6. Safi, A.; Singh, S. A systematic literature review on phishing website detection techniques. *J. King Saud Univ. Comput. Inf. Sci.* **2023**, *35*, 590–611. [CrossRef]
7. Google. *Overview | Safe Browsing APIs (v4)*; Google: Mountain View, CA, USA, 2024.
8. Li, W.; Laghari, S.U.A.; Manickam, S.; Chong, Y.-W.; Li, B. Machine Learning-Enabled Attacks on Anti-Phishing Blacklists. *IEEE Access* **2024**, *12*, 191586–191602. [CrossRef]
9. Barraclough, P.A.; Fehringer, G.; Woodward, J. Intelligent cyber-phishing detection for online. *Comput. Secur.* **2021**, *104*, 102123. [CrossRef]
10. Gupta, B.B.; Yadav, K.; Razzak, I.; Psannis, K.; Castiglione, A.; Chang, X. A novel approach for phishing URLs detection using lexical based machine learning in a real-time environment. *Comput. Commun.* **2021**, *175*, 47–57. [CrossRef]
11. Al-Sarem, M.; Saeed, F.; Al-Mekhlafi, Z.G.; Mohammed, B.A.; Al-Hadhrami, T.; Alshammari, M.T.; Alreshidi, A.; Alshammari, T.S. An optimized stacking ensemble model for phishing websites detection. *Electronics* **2021**, *10*, 1285. [CrossRef]
12. Foozy, C.F.M.; Anuar, M.A.I.; Maslan, A.; Adam, H.A.M.; Mahdin, H. Phishing URLs Detection Using Naives Baiyes, Random Forest and LightGBM Algorithms. *Int. J. Data Sci.* **2024**, *5*, 56–63. [CrossRef]
13. Vanitha, N.; Vinodhini, V. Malicious-URL detection using logistic regression technique. *Int. J. Eng. Manag. Res. IJEMR* **2019**, *9*, 108–113.
14. Nowroozi, E.; Mohammadi, M.; Conti, M. An adversarial attack analysis on malicious advertisement URL detection framework. *IEEE Trans. Netw. Serv. Manag.* **2022**, *20*, 1332–1344. [CrossRef]
15. Pham, D.M.; Shao, Y.; Li, Z.; Makanju, A.; Alomari, Z. Network Intrusion Detection with CNNs: A Comparative Study of Deep Learning and Machine Learning Models. In *Proceedings of the 2024 2nd International Conference on Computer, Vision and Intelligent Technology (ICCVIT)*; IEEE: New York, NY, USA, 2024; pp. 1–6.
16. Gupta, B.; Gaurav, A.; Attar, R.; Arya, V.; Alhomoud, A.; Chui, K.T. Optimized phishing detection with recurrent neural network and whale optimizer algorithm. *Comput. Mater. Contin.* **2024**, *80*, 4895. [CrossRef]
17. Xiao, X.; Zhang, D.; Hu, G.; Jiang, Y.; Xia, S. CNN–MHSA: A Convolutional Neural Network and multi-head self-attention combined approach for detecting phishing websites. *Neural Netw.* **2020**, *125*, 303–312. [CrossRef] [PubMed]
18. Le, H.; Pham, Q.; Sahoo, D.; Hoi, S.C.H. URLNet: Learning a URL representation with deep learning for malicious URL detection. *arXiv* **2018**, arXiv:1802.03162. [CrossRef]
19. Kamran, S.A.; Sengupta, S.; Tavakkoli, A. Semi-supervised conditional GAN for simultaneous generation and detection of phishing URLs: A game theoretic perspective. *arXiv* **2021**, arXiv:2108.01852.
20. Liang, Y.; Deng, J.; Cui, B. Bidirectional LSTM: An innovative approach for phishing URL Identification. In *Proceedings of the International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*; Springer International Publishing: Cham, Switzerland, 2019; pp. 326–337.
21. Maneriker, P.; Stokes, J.W.; Lazo, E.G.; Carutasu, D.; Tajaddodianfar, F.; Gururajan, A. URLtran: Improving phishing URL detection using transformers. In *Proceedings of the Milcom 2021–2021 IEEE Military Communications Conference (Milcom)*; IEEE: New York, NY, USA, 2021; pp. 197–204.
22. Liu, R.; Wang, Y.; Guo, Z.; Xu, H.; Qin, Z.; Ma, W.; Zhang, F. TransURL: Improving malicious URL detection with multi-layer Transformer encoding and multi-scale pyramid features. *Comput. Netw.* **2024**, *253*, 110707. [CrossRef]
23. Liu, R.; Wang, Y.; Xu, H.; Qin, Z.; Zhang, F.; Liu, Y.; Cao, Z. PMANet: Malicious URL detection via post-trained language model guided multi-level feature attention network. *Inf. Fusion* **2025**, *113*, 102638. [CrossRef]
24. Rabbani, M.; Eskandarian, M.; Mirani, M.; Piya, G.; Opushnyev, I.; Lu, R.; Dadkhah, S. URL2Path: A Robust Graph Learning Approach for Malicious URL Detection. *IEEE Trans. Reliab.* **2026**, *75*, 1145–1159. [CrossRef]
25. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*; Curran Associates Inc.: Red Hook, NY, USA, 2017; Volume 30.
26. Lin, T.; Wang, Y.; Liu, X.; Qiu, X. A survey of transformers. *AI Open* **2022**, *3*, 111–132. [CrossRef]
27. Wang, J.; Li, C.; Li, L.; Huang, Z.; Wang, C.; Zhang, H.; Zhang, Z. InSAR time-series deformation forecasting surrounding Salt Lake using deep transformer models. *Sci. Total Environ.* **2023**, *858*, 159744. [CrossRef]
28. Kiranyaz, S.; Avci, O.; Abdeljaber, O.; Ince, T.; Gabbouj, M.; Inman, D.J. 1D convolutional neural networks and applications: A survey. *Mech. Syst. Signal Process.* **2021**, *151*, 107398. [CrossRef]
29. Haq, Q.E.; Faheem, M.H.; Ahmad, I. Detecting phishing URLs based on a deep learning approach to prevent cyber-attacks. *Appl. Sci.* **2024**, *14*, 10086. [CrossRef]

30. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 2–9 February 2021; Volume 35, pp. 11106–11115.
31. Elsadig, M.; Ibrahim, A.O.; Basheer, S.; Alohali, M.A.; Alshunaifi, S.; Alqahtani, H.; Alharbi, N.; Nagmeldin, W. Intelligent deep machine learning cyber phishing URL detection based on bert features extraction. *Electronics* **2022**, *11*, 3647. [CrossRef]
32. Trivedi, P.; Heimann, M.; Koutra, D.; Thiagarajan, J.J. Interrogating Paradigms in Self-supervised Graph Representation Learning. In Proceedings of the Tenth International Conference on Learning Representations, Virtual, 25–29 April 2021.
33. Motyka, V.; Vysotska, V.; Chyrun, L.; Markiv, O.; Chyrun, S.; Kolyasa, L. System project for Ukrainian-language feedback tonality analysis in the health care field based on BERT model. In *Proceedings of the 2023 IEEE 18th International Conference on Computer Science and Information Technologies (CSIT)*; IEEE: New York, NY, USA, 2023; pp. 1–6.
34. Almujahid, N.F.; Haq, M.A.; Alshehri, M. Comparative evaluation of machine learning algorithms for phishing site detection. *PeerJ Comput. Sci.* **2024**, *10*, e2131. [CrossRef]
35. Opara, C.; Chen, Y.; Wei, B. Look before you leap: Detecting phishing web pages by exploiting raw URL and HTML characteristics. *Expert Syst. Appl.* **2024**, *236*, 121183. [CrossRef]
36. Gomes de Barros, J.C.; Revoredo da Silva, C.M.; Candeia Teixeira, L.; Fernandes, B.J.T.; de Oliveira, J.F.L.; Feitosa, E.L.; Santos, W.P.D.; Arcoverde, H.F.; Garcia, V.C. Piracema: A Phishing snapshot database for building dataset features. *Sci. Rep.* **2022**, *12*, 15149. [CrossRef]

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.