A Hybrid Deep Learning Framework for Multi-Class Malicious URL in QR-Code Detection

LUKMAN ADEBAYO OGUNDELE[1]
JULIUS TEMITAYO ADEPOJU[2]
FEMI EMMANUEL AYO[1]
IDAYAT ABIKE AKANO[3]
OLUYEMISI ADENIKE OYEDEMI[3]
FASILAT ADERIBIGBE[4]

[1] Department of Computer Sciences, Faculty of Science, Olabisi Onabanjo University, Ago-Iwoye, Ogun State, Nigeria
[2] Department of Mathematical Science, Faculty of Physical Science, University of Ilorin, Kwara State, Nigeria
[3] Department of Cyber-security, Faculty of Computing, University of Ilesa, Ilesa, Osun State, Nigeria
[4] Department of Computer Science, Faculty of Communication and Information Science, University of Ilorin, Nigeria
[1]ogundele.lukman@oouagoiwoye.edu.ng, ayo.femi@oouagoiwoye.edu.ng
[2]adepojujulius58@gmail.com
[3]idayat_akano@unilesa.edu.ng, yemisi.oyedemi@unilesa.edu.ng
[4]shuaib.fy@unilorin.edu.ng

Abstract. The proliferation of Quick Response (QR) codes in mobile payments, authentication, and marketing has created a new attack surface: cybercriminals now embed malicious URLs inside otherwise ordinary-looking QR codes to redirect unsuspecting users to phishing, malware, and defacement pages. Conventional detection techniques such as blacklisting and heuristic filtering are unable to keep pace with these dynamically evolving schemes, motivating the need for a sophisticated neural network-based approach. This study proposes a hybrid deep learning framework, ResCNN-BiLSTM (Residual Convolutional Neural Network combined with a Bidirectional Long Short-Term Memory network and an attention mechanism), for classifying URLs decoded from QR-code images into four categories: Benign, Malware, Defacement, and Phishing. URLs from the ISCX URL 2016 corpus were first rendered as QR-code images and subsequently decoded back into text, so that the model is evaluated on the exact character sequence a scanning device would recover from a QR code, while lexical and statistical features are additionally engineered from this decoded string. Optimized feature selection and extraction mechanisms are incorporated to improve classification accuracy. The performance of the model is evaluated using precision, recall, F1-score, and Area Under the Curve (AUC). The model effectively identifies Malware and Defacement URLs, with AUC values of 0.99 and 0.98, respectively. However, phishing URL detection remains more challenging, with a lower AUC of 0.92 and F1-score of 0.74; Section 4.3 discusses the practical implications of this precision/recall trade-off in more depth. This research reaffirms the efficiency of deep learning algorithms, and specifically ResCNN-BiLSTM, for QR-code-embedded URL classification, while identifying phishing detection as the primary area requiring further refinement. Future work is proposed to combine ResCNN-BiLSTM with adversarial training and broader cross-study benchmarking to strengthen robustness and comparability.

Keywords: QR codes, Neural Network, CNN, BiLSTM, Malicious URL Detection

(Received July 19, 2025 / Accepted August 3, 2026)

1 Introduction

A vital component of contemporary digital interactions, QR codes facilitate easy access to information, payments, and authentication procedures [41, 42]. Their extensive use in marketing, healthcare, and finance has drawn cybercriminals, who use them to spread malicious URLs [16]. Robust detection mechanisms [33, 32, 43, 31, 29, 48, 30, 26] are necessary to protect users from phishing websites, malware-infected pages, and fraudulent portals that harvest sensitive data. These attacks create misleading QR codes that lead users to these sites [16]. Rule-based filtering, blacklisting, and heuristic analysis are examples of traditional security techniques that are at odds with zero-day attacks and dynamically generated URLs that use link shorteners, obfuscation, and domain generation algorithms (DGAs) to avoid detection [5, 4, 7, 6]. Because of their capacity to identify patterns in large datasets and extrapolate to threats that haven't been identified yet, machine learning (ML) and deep learning (DL) techniques have become increasingly popular [8, 9].

Convolutional Neural Networks (CNNs) are ideally suited for examining the structure and patterns found in URLs embedded in QR codes because of their impressive performance in extracting spatial and hierarchical features [45, 46]. In mathematical terms, a CNN model can be expressed as a function $f_\theta(X)$, where X represents the input QR code features and θ represents the learnable parameters. The objective is to minimize a loss function $L(y, f_\theta(X))$ by training with optimization algorithms such as Stochastic Gradient Descent (SGD) [1, 46, 47]. In addition to more sophisticated methods like TF-IDF vectorization, word embeddings, and attention mechanisms to concentrate on crucial malicious components, feature extraction is essential and involves both conventional URL-based lexical analysis, domain reputation, and HTTP behavior [22, 19, 2, 21]. Having well-balanced datasets of legitimate, phishing, and malware-infected QR codes is necessary for these models to be effective. URL transformation and adversarial training may be added to increase resilience [21, 37].

It is important to clarify, at the outset, exactly what "QR code" means in the context of this study's classification pipeline. A QR code is fundamentally a lossless, machine-readable encoding of a text string; once a QR-code image is decoded (e.g., via a standard QR reader), the original URL string is recovered byte-for-byte. Consequently, the security-relevant signal for detecting malicious QR-embedded URLs lies in the decoded URL string itself, not in the visual layout of the QR-code matrix, since two QR codes encoding the same URL are visually near-identical while two QR codes encoding different URLs (one benign, one malicious) can be visually indistinguishable from a human perspective but textually very different. This is the design rationale that underlies the modelling choices described in Section 3, and it is stated explicitly here to avoid the ambiguity raised during review regarding whether classification operates on QR pixel data or on the decoded URL sequence.

Lack of user awareness is a major problem since many users scan QR codes without confirming their legitimacy, leaving them vulnerable to attacks [25]. Real-time protection can be achieved by incorporating CNN-based detection into browser extensions and mobile security apps, but doing so necessitates minimizing computational overhead and inference time [3]. To enhance capabilities, hybrid models that combine CNNs with Transformers or Long Short-Term Memory (LSTM) networks can be investigated; Transformers use self-attention mechanisms to identify subtle malicious patterns, while LSTMs capture sequential dependencies in URLs [39, 28, 15, 17]. Furthermore, it is important to ensure adversarial robustness because attackers might alter URLs to trick the model; strategies like robust feature selection, generative adversarial networks (GANs), and adversarial training can reduce these risks [3].

Building a sophisticated deep learning-based system for identifying malicious URLs with a Residual Convolutional Neural Network (ResNet) is the main goal of this research in order to increase the precision and resilience of URL classification. To classify URLs into benign, defacement, phishing, and malware categories, a residual CNN-based model must be designed and implemented. Optimal feature selection and extraction techniques must be incorporated, and the model must be assessed using metrics such as accuracy, precision, recall, F1-score, and AUC-ROC; to compare its performance with other existing deep learning techniques like conventional CNNs and LSTMs; to boost its generalization ability to handle different real-world datasets; and to experiment with the hybrid Residual CNN+RNN architecture to better handle sequential information. The significance of this research is that it presents an intelligent and adaptive security mechanism that helps protect users and businesses from QR code-related phishing attacks and malware, thus creating a safer and more trustworthy online environment in a QR code-infused world.

2 Related Work

The emergence of malicious URLs has become a significant cybersecurity issue, requiring sophisticated detection methods using machine learning (ML) and

deep learning (DL). Conventional rule-based and signature-based detection methods are less effective against dynamic attack patterns, leading researchers to investigate innovative techniques for enhancing the accuracy of malicious URL detection. Recently, various ML and DL-based frameworks, such as ensemble learning, transformer networks, behavioral analysis, and multi-field relation mining, have been proposed by researchers for malicious URL detection.

[10] presented mURLi, a detection system capable of detecting malicious URLs, SQL injection (SQLi), and NoSQL injection (NoSQLi) attacks. The authors conducted an experiment to compare the performance of various ML and DL algorithms, such as Random Forest (RF), XGBoost, AdaBoost, K-Nearest Neighbors (KNN), Artificial Neural Networks (ANN), Bidirectional Long Short-Term Memory (BiLSTM), and Bidirectional Encoder Representations from Transformers (BERT). The experimental outcome showed that BERT provided the best accuracy (99.6%) for SQLi detection and 99.01% for NoSQLi detection because of its superior language understanding capabilities. BiLSTM performed better than other algorithms for malicious URL detection with an accuracy of 95.2%, indicating its superiority in sequence-based URL classification.

[24] developed PMANet, which is based on pre-trained language model guidance and multi-level feature attention networks, for improving malicious URL detection. The authors found that existing state-of-the-art solutions based on the transformer architecture had some drawbacks, such as domain adaptability and the absence of character-level features. Therefore, PMANet integrated a post-training approach for URL data, using three self-supervised tasks such as masked language modeling, noisy language modeling, and domain discrimination. PMANet showed better performance in handling adversarial attacks, achieving an AUC score of 0.9941 while detecting active malicious URLs, outperforming existing deep learning techniques.

[44] developed AutoHTTP, an attention mechanism-based model for detecting malicious HTTP traffic. Unlike existing solutions, AutoHTTP used multi-field relation mining to analyze different components of HTTP requests, such as user-agent, URLs, and requests. AutoHTTP integrated an elementary feature extraction module for transforming raw data from networks into structured data. It used an attention and cross-network mechanism to identify semantic correlations among different fields, showing better performance over existing feature-engineered solutions.

[20] examined the analysis of behavioral patterns for malicious URL detection, targeting habitual manipulation patterns of attackers. The authors analyzed more than 1.5 million malicious URLs gathered over two years, detecting habitual adversarial patterns. By applying similarity matching, the method proposed a URL classification scheme according to behavioral patterns of attackers. The proposed model showed reasonable accuracy (up to 70%) and could be used as a lightweight web filtering system to pre-filter potentially malicious URLs. This behavioral pattern-based method offered an alternative to the conventional ML-based approach, targeting attack pattern analysis instead of direct feature extraction.

These studies together illustrate the development of malicious URL detection methods from conventional ML-based approaches to transformer-based and attention-driven models. Although ensemble learning and BiLSTM-based recurrent models have shown excellent performance, transformer models such as BERT and PMANet offer better accuracy by applying deep contextual understanding and hierarchical feature extraction. Moreover, AutoHTTP and behavioral pattern analysis offer new insights into network security by exploiting multi-field correlations and adversarial patterns. Future studies should aim to combine these methods into real-time security frameworks, addressing issues such as adversarial robustness, cross-domain generalization, and scalability in large-scale cybersecurity systems.

[18] proposed a deep learning approach based on Gated Recurrent Unit and Artificial Fish Swarm Algorithm for URL classification into dangerous categories. The proposed approach, AFSADL-MURLC, maximizes the efficiency of the GRU model using an artificial fish swarm algorithm and Glove word embedding. The proposed approach was evaluated using a Kaggle dataset, showing that the proposed approach outperforms other approaches. However, an evaluation of the approach based on hostile attacks that may change URLs to evade detection is not included in this research.

[27] proposed an ensemble learning approach for detecting malicious URLs, named SeizeMaliciousURL. This approach uses multiple classifiers and combines

their results to improve classification accuracy. This approach uses filtering based on thresholding to improve classification accuracy. However, this approach does not take into account real-time mutation of URLs, which attackers often use to evade detection.

[35] proposed an approach to evaluate adversarial attacks on deep learning approaches for detecting malicious URLs. This research showed that CNN, used for URL classification, can be manipulated by attackers, which reduces its accuracy by more than 56%. They proposed adversarial training to enhance robustness, limiting the accuracy drop to less than 7%. Despite this improvement, the study highlights the ongoing challenge of securing DL-based models against adversarial threats.

[22] specifically focused on feature engineering techniques for improving malicious URL detection. A combination of linear and nonlinear space transformation techniques, such as singular value decomposition and the Nystrom method, was used for improving k-Nearest Neighbor, Support Vector Machine, and Multi-Layer Perceptron model identification. However, this technique is based on manual feature selection, which may not be suitable for adapting to changing malicious URL detection schemes.

The above research studies collectively outline improvements to malicious URL detection using ML, DL, and feature engineering techniques. Although DL techniques show good performance, their vulnerability to adversarial attacks is still an open issue. Feature engineering techniques improve classical ML model performance but may not be suitable for adapting to new attack schemes. Therefore, researchers should focus on developing hybrid models that incorporate DL, feature selection, and adversarial attack detection to improve performance.

Phishing and malicious URL detection is an important research area due to its evolving nature. Researchers have proposed different deep learning, machine learning, and optimization techniques for improving detection accuracy and efficiency.

[11] presented an integrated phishing URL detection model based on a combination of deep learning classifiers and a pre-trained transformer model. The model is designed to analyze both structured and unstructured URL sequences. The model is based on a combination of RasNet (ResNet-based feature extractor), TCMA (Temporal Convolutional Network with Multi-Head Self-Attention), and MPNet (Masked and Permuted Pre-training for Language Understanding). The model has been tested against several datasets and has achieved a 99.71% detection rate. The model has outperformed other existing models by achieving an improvement of 1.37% to 2.01%.

[34] presented a malicious URL detection approach based on a combination of a learned bloom filter and an evolutionary convolutional neural network. The approach has been named deepBF. The approach has introduced a non-cryptographic string hash function to filter malicious URLs. The approach has been efficient in detecting malicious URLs while the filter is dynamically updated based on new URLs. The results of the approach have proved the efficiency of the approach in detecting malicious URLs.

[11] further expanded their work by using a Temporal Convolutional Network (TCN) and Multi-Head Self-Attention (MHSA) for malicious URL detection. Their proposed model overcame CNN's inability to process non-spatial data and RNN's inability to learn long-term dependencies. Their model utilized both character-level and word-level embeddings to enhance classification accuracy, reaching 98.78% accuracy, outperforming traditional methods.

[40] aimed to optimize the detection of malicious websites by utilizing the Firefly algorithm for feature selection and a Particle Swarm Optimization (PSO)-optimized XGBoost classifier. Their proposed method was tested on a collection of over 36,000 websites, resulting in a classification accuracy of 98.42% for binary classification and over 98% accuracy for multi-class classification, proving the efficacy of their proposed model.

[36] investigated the use of Large Language Models (LLMs) for phishing URL classification. They proposed a one-shot learning approach utilizing LLMs, incorporating Chain-of-Thought reasoning for enhanced classification and interpretability. Their experiment on three different datasets revealed that GPT-4 Turbo attained an F1 score of 0.92 for one-shot classification, offering comparable results to supervised learning models while providing human-interpretable explanations.

[14] designed a CNN-based URL phishing detection framework with the incorporation of Brown-Bear

Optimization (BBOA) for hyperparameter optimization. The proposed framework utilized Random Forest for feature extraction and was tested on a Kaggle dataset consisting of 11,000+ website URLs. The proposed method reported a classification accuracy of 93% and a precision of 92%, proving the efficacy of optimization algorithms in improving phishing detection models.

The above-mentioned research works showcase the progress in phishing and malicious URL detection, proving the efficacy of deep learning models, transformers, optimization algorithms, and LLMs in improving phishing detection models. The combination of innovative models like TCN-MHSA, ResNet, and BBOA, along with adaptive filtering methods such as deepBF and Bloom Filters, is continuously breaking the barriers in cybersecurity solutions for combating threats.

Table 1: Summary of Related Works

Author	Method	Dataset	Result	Limitation of the Study
[10]	mURLi tool using ML & DL models (RF, XGBoost, AdaBoost, KNN, ANN, BiLSTM, BERT)	Not specified	BERT achieved 99.6% accuracy for SQLi, 99.01% for NoSQLi, and BiLSTM reached 95.2% accuracy for malicious URLs	No dataset details provided; lacks real-time attack adaptation
[23]	PMANet (pre-trained transformer with multi-level feature attention)	Real-world malicious URL datasets	AUC of 0.9941 under adversarial attacks; detected all 20 active malicious URLs	High computational cost due to transformer-based processing
[44]	AutoHTTP (multi-field relation mining using attention and cross-network)	CTU-13, CICAndMal, ISCX-URL	Outperformed feature-based models in HTTP traffic analysis	Focuses only on HTTP-based attacks; lacks generalization for other URL types
[20]	Behavioral analysis with similarity matching for URL classification	1,529,433 malicious URLs	Achieved up to 70% accuracy in detecting habitual attack patterns	Lower accuracy compared to ML/DL models; not effective for zero-day attacks
[18]	Artificial Fish Swarm Algorithm (AFSA) + Gated Recurrent Unit (GRU)	Kaggle malicious URL dataset	High accuracy in detecting malicious URLs	Lacks robustness against adversarial attacks
[27]	Ensemble learning-based classifier aggregation	Malicious URL datasets	Improved classification accuracy over traditional ML models	Does not consider real-time URL mutation by attackers
[35]	Adversarial attack evaluation on deep learning-based URL classifiers	Multiple CNN-based URL detection models	Demonstrated susceptibility of CNN classifiers to adversarial manipulation	High accuracy drop under adversarial attacks
[22]	Feature engineering (linear and nonlinear transformations)	331,622 URLs with 62 features	Improved accuracy of k-NN, SVM, and MLP classifiers	Manual feature selection may not generalize to evolving threats
[11]	RasNet (ResNet-based feature extractor) + TCMA (Temporal CNN with Multi-Head Self-Attention) + MPNet	Multiple datasets	Achieved 99.71% accuracy, outperforming existing models by 1.37% to 2.01%	Requires high computational resources due to deep learning complexity
[34]	deepBF (Learned Bloom Filter + Evolutionary CNN)	Malicious URL datasets	Efficient filtering and high detection accuracy	Filtering updates may introduce latency in real-time detection
[12]	Temporal CNN + Multi-Head Self-Attention (MHSA)	Multiple datasets	98.78% accuracy, outperforming CNN and RNN-based models	Complexity in processing long-sequence URL data
[40]	Firefly Algorithm for feature selection + PSO-optimized XGBoost	36,000+ websites	98.42% accuracy in binary classification; >98% in multi-class classification	Dependence on the quality of feature selection for optimal performance
[36]	Large Language Model (LLM) with one-shot learning and Chain-of-Thought reasoning	Three phishing URL datasets	GPT-4 Turbo achieved F1-score of 0.92 in one-shot setting	High inference cost and dependency on pre-trained LLMs
[13]	CNN-based URL detection + Brown-Bear Optimization (BBOA) + Random Forest for feature selection	Kaggle dataset (11,000+ URLs)	93% accuracy, 92% precision	Moderate accuracy compared to transformer-based approaches
This study	ResCNN-BiLSTM: QR-code encoding/decoding pipeline + Residual CNN feature extractor + attention-augmented BiLSTM classifier	ISCX URL 2016 (651,191 URLs rendered/decoded as QR codes)	AUC of 0.99 (Malware), 0.98 (Defacement), 0.96 (Benign), 0.92 (Phishing)	Phishing class remains harder to separate from Benign; benchmarked against a single external lightweight-DL baseline (Section 4.4)

3 Materials and Methods

This chapter describes the design and implementation of the malicious URL detection system. The system combines the concepts of deep learning, specifically a Residual CNN and a Recurrent Neural Network (RNN), to classify URLs into various categories, such as benign, phishing, malware, and defacement. The process involves several important steps: preprocessing, feature extraction, classification, and evaluation. The proposed system utilizes both the character-level representation of URLs and numerical features, thereby providing a holistic analysis of malicious URLs.

3.1 System Architecture for Malicious URL Classification

The system is intended to be efficient in processing and classifying URLs using a hybrid deep learning approach that combines the use of Residual Convolutional Neural Networks (ResNet) for feature extraction and a Bidirectional Long Short-Term Memory (BiLSTM) network with an attention mechanism for classification. The system has a well-structured pipeline that comprises four major components:

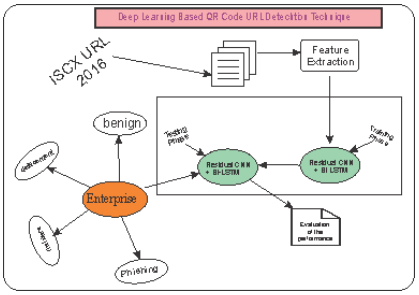

Figure 1: QR Code Detection System. The pipeline begins with the ISCX URL 2016 corpus of raw URL strings; each URL is programmatically rendered into a QR-code image and then decoded back to text (Section 3.2), so the classifier consumes the exact character sequence a mobile scanner would recover from the QR code, in addition to lexical/statistical features engineered from that decoded string (Section 3.4).

3.2 QR Code Generation and Decoding Pipeline

All 651,191 URL instances from the merged data set (Section 3.3) have been automatically converted to a QR-code image by means of the conventional QR code generation algorithm (similar to the commonly used python library qrcode), which employs error-correction level M and a fixed module size, such that all URLs, whether benign, phishing, malware, or defacement, are represented as a QR-code image. Following this, each of the QR-code images was decoded to the corresponding URL instance by a QR decoder, thereby extracting the same information that a typical smart-phone camera or QR reader would obtain when decoding. The above encode-decode operation accomplishes two objectives: (i) it confirms the validity of the QR-code images obtained and their information-preserving nature, and (ii) it makes sure that the classification problem is the same as classifying the URL string extracted by QR decoding.

Since the equivalence is already proven, the subsequent steps of feature extraction and sequence modeling (Sections 3.4 and 3.5) are performed based on the sequence of characters derived through URL decoding from the QR code image instead of QR pixel data directly. That is how the issue of Algorithm 1 gets resolved: the Conv1D layers of the Residual CNN block work with the one-dimensional sequence of tokenized/embedded characters that were obtained after QR decoding, and not with two-dimensional QR pixel data. The two-dimensional convolution (Conv2D) could have been used only in case when the classifier was trained on raw QR images (for example, to detect QR tampering or spoofing) which corresponds to another problem beyond the scope of the current work.

3.3 QR Code URL Detection

URL detection is an essential stage in detecting harmful URLs that may potentially cause security attacks like phishing, malware, or website defacement. During this stage, the collection of raw URLs from various sources was carried out. For the creation of an effective dataset, various open-source datasets like ISCX URL 2016, Malware Domain Blacklist, Faizan Git Repository, PhishTank, and PhishStorm were used to collect URLs. These datasets are useful in providing valuable information regarding URLs, which helps in the creation of a diverse set of URLs like benign, phishing, malware, or defacement URLs.

After collecting the URLs, preprocessing was carried out to refine the collected URLs. During this stage, various operations like eliminating redundant information, special characters, or unnecessary information like URL encoding or tracking parameters were removed from the URLs. Tokenization operations were also used to break down the URLs to various components like domains, subdomains, or query parameters. As mentioned in section 3.2, this preprocessing technique was used for the URL string, which was obtained after the QR code was decoded, to ensure that the features are based on realistic data from the scanner.

The breakdown of URLs helped in analyzing the URLs more effectively to extract useful information.

3.4 ISCX URL 2016 Dataset

A comprehensive dataset of 651,191 URLs was developed, with 428,103 URLs classified as benign, 96,457 as defacement, 94,111 as phishing, and 32,520 as malware URLs. Due to the vast and varied nature of the dataset, it was divided into individual data frames based on their sources before being merged into a single dataset. Each URL in the merged dataset was then encoded into, and decoded from, a QR-code image as described in Section 3.2, prior to any further preprocessing. The dataset was then split into training, validation, and testing sets, with each set representing 70%, 15%, and 15% of the total dataset, respectively. The training dataset was used for the development of the classification model, the validation dataset for model selection, and the testing dataset for model evaluation. This systematic approach ensured that the dataset was well balanced, eliminating any form of bias and overfitting, thereby improving the effectiveness of the detection model [38].

3.5 Preprocessing and Feature Engineering

Before detection, raw URLs (recovered from QR-code decoding) are subjected to preprocessing to extract significant features and represent them in a numerical form. The preprocessing techniques involve:

3.5.1 Tokenization

URLs are composed of various elements, including domain, subdomain, path, and query parameters. Tokenization is the process of decomposing URLs into significant parts, including:

- Protocol: Discriminating between HTTP, HTTPS, and FTP.
- Domain and Subdomain: Identifying the main domain and subdomains.
- Path and Query Parameters: Examining other URL parts that could represent malicious activity.

3.5.2 Feature Extraction

The important lexical and statistical properties are extracted from the URLs, which are:

- URL Length: Malicious URLs are generally longer.
- Digit Count: A high digit count can be an indication of automated URL creation.
- Number of Subdomains: Too many subdomains can be a sign of phishing attacks.
- Special Characters: Characters such as '@', '-', and '
- Entropy Score: It calculates the randomness of the URL, and a high score indicates that the URL is trying to hide something.
- HTTPS Usage: Secure sites use HTTPS.

3.6 Model Architecture

This model integrates Residual Convolutional Neural Networks (ResCNN) with Bidirectional Long Short-Term Memory (BiLSTM) and an attention layer to increase classification accuracy. The Residual CNN helps to recognize spatial patterns in the data, while the BiLSTM helps to recognize sequential dependencies in the data. By integrating these two components, the model increases its ability to recognize features in the data and to recognize sequential dependencies in the data, making it useful for *malware detection, time series classification, and URL threat analysis*. The mathematical formulation is

1. Residual CNN Block: A Residual Block is defined as:

$$\mathbf{X}_l = F(\mathbf{X}_{l-1}, W_l) + \mathbf{X}_{l-1} \tag{1}$$

where:

- $\mathbf{X}_{l-1}$ is the input to layer l (the embedded, decoded URL character/token sequence recovered after QR-code decoding, per Section 3.2).
- $F(\mathbf{X}_{l-1}, W_l)$ represents the transformation (convolution, batch normalization, activation).
- W_l are the learnable weights of the CNN filters.
- $+\mathbf{X}_{l-1}$ is the skip connection, ensuring gradient propagation.

Each convolutional layer applies a 1D convolution followed by batch normalization:

$$F(\mathbf{X}) = \sigma(\text{BN}(\mathbf{W} * \mathbf{X})) \tag{2}$$

where:

- $*$ denotes the convolution operation (1D, applied along the decoded URL token sequence – see Section 3.2 for the justification of Conv1D over Conv2D).
- $\sigma(\cdot)$ is the ReLU activation function.
- Batch Normalization (BN) improves convergence speed.

2. Max-Pooling Layer: To reduce the feature map size and retain dominant features, we apply 1D Max-Pooling:

$$\mathbf{X}_i^{\text{pooled}} = \max_{j \in \{1,\ldots,k\}} \mathbf{X}_{i+j} \tag{3}$$

where k is the pooling window size.

3. Bidirectional LSTM Layer: A standard LSTM unit at time t computes:

$$\mathbf{f}_t = \sigma(W_f \mathbf{h}_{t-1} + U_f \mathbf{x}_t + b_f) \tag{4}$$

$$\mathbf{i}_t = \sigma(W_i \mathbf{h}_{t-1} + U_i \mathbf{x}_t + b_i) \tag{5}$$

$$\mathbf{o}_t = \sigma(W_o \mathbf{h}_{t-1} + U_o \mathbf{x}_t + b_o) \tag{6}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(W_c \mathbf{h}_{t-1} + U_c \mathbf{x}_t + b_c) \tag{7}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{8}$$

where:

- $\mathbf{f}_t$ = forget gate, $\mathbf{i}_t$ = input gate, $\mathbf{o}_t$ = output gate.
- $\mathbf{c}_t$ = cell state, $\mathbf{h}_t$ = hidden state.
- W, U, b are learnable parameters.
- $\odot$ denotes element-wise multiplication.

In Bidirectional LSTM (BiLSTM), two LSTMs process the input forward and backward, concatenating hidden states:

$$\mathbf{h}_t^{\text{bi}} = [\mathbf{h}_t^{\text{forward}}; \mathbf{h}_t^{\text{backward}}] \tag{9}$$

This enables the model to learn from both past and future information.

4. Attention Mechanism: Additive attention Bahdanau (as referred to as the Bahdanau-type) is employed on top of the BiLSTM hidden state representations $h1^b i, ..., h_T^b i$. This allows the system to allocate more weight to those characters which are more informative with regard to the presence of maliciousness (such as unusual tokens, subdomains, or entropy).

$$e_t = \mathbf{v}^\top \tanh(W_a \mathbf{h}_t^{\text{bi}} + b_a) \tag{10}$$

$$\alpha_t = \frac{\exp(e_t)}{\sum_{k=1}^{T} \exp(e_k)} \tag{11}$$

$$\mathbf{c} = \sum_{t=1}^{T} \alpha_t \, \mathbf{h}_t^{\text{bi}} \tag{12}$$

here, W_a and v refer to learnable attention weights, α_t refers to the normalized weight of attention for time-step t, and $\mathbf{c}$ refers to the resultant context vector that represents the sequence, emphasizing the most discriminative parts. The context vector $\mathbf{c}$, and not just the BiLSTM hidden state vector at the end of the sequence, becomes the input to the fully connected classification head thereafter. Essentially, this process helps most in differentiating between Phishing and Benign URLs, since it allows the network to focus on suspicious substrings in the URL even if the length and overall structure of the URL looks similar to a genuine URL; however, as pointed out in Section 4.3, this class pair is the toughest to separate.

5. Fully Connected Layers & Softmax Output: A fully connected layer maps the attention context vector $\mathbf{c}$ to class probabilities:

$$\mathbf{y} = \text{softmax}(\mathbf{W}\mathbf{c} + b) \tag{13}$$

where the softmax function ensures:

$$y_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \tag{14}$$

for C classes.

6. Model Training & Optimization We train the model using the Adam optimizer and categorical cross-entropy loss function:

$$\mathcal{L} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{15}$$

where y_i is the true label and $\hat{y}_i$ is the predicted probability.

7. Advantages of this Hybrid Model

   - Better Feature Extraction: CNN captures spatial dependencies, while LSTM captures sequential dependencies.
   - Gradient Flow Stability: Residual connections prevent vanishing gradients.
   - Better Temporal Learning: BiLSTM learns past and future dependencies.
   - Focused Discrimination: the attention layer lets the classifier emphasize the most suspicious substrings of a URL rather than weighting every character equally.
   - Generalization: Dropout and batch normalization reduce overfitting.

This Residual CNN + BiLSTM with attention hybrid model effectively leverages both CNN and RNN for feature extraction and sequential learning, particularly for tasks such as *malware classification, anomaly detection, and QR code URL detection*. The mathematical formulations guarantee robust performance.

3.7 Implementation Details and Hyperparameters

Table 2 summarizes the architecture configuration and training parameters used to obtain the results reported in Section 4.

Table 2: Architecture and training hyperparameters

Parameter	Value
Residual block 1 filters / kernel size	64 / 3
Residual block 2 filters / kernel size	128 / 3
Max-pooling window (k)	2
BiLSTM layer 1 units	128
BiLSTM layer 2 units	64
Attention dimensionality	64
Dense layer sizes	128, 64
Dropout rate	0.3
Optimizer	Adam
Learning rate (initial)	1×10^{-3}
Loss function	Sparse categorical cross-entropy
Batch size	64
Maximum epochs	50
Early stopping patience	5 (validation loss)
LR reduction factor / patience	0.5 / 3
Train / validation / test split	70% / 15% / 15%

3.8 Performance Evaluation

The model's effectiveness is evaluated using standard performance metrics and visualization techniques.

Algorithm 1 Residual CNN and Bi-LSTM with Attention for Malicious URL Classification (operating on the decoded URL sequence recovered from the QR-code, per Section 3.2)

Require: QR-decoded URL features X with labels Y
Ensure: Trained model for URL classification

1: **Data Preprocessing:** Reshape input X to $(samples, features, 1)$ – one feature dimension per decoded character/token position, i.e., a 1D sequence, which is why Conv1D (not Conv2D) is used below.
2: **Initialize Model:** Define input layer with shape $(sequence\_length, 1)$.
3: **Residual CNN Layers:**
4: Apply Conv1D $\times 2$ with Batch Normalization and ReLU activation.
5: Add a residual connection with Conv1D (kernel size 1).
6: Apply MaxPooling1D.
7: Repeat for the second residual block with 128 filters.
8: **Bi-LSTM Layers:**
9: Apply two Bidirectional LSTM layers with 128 and 64 units.
10: **Attention Layer:**
11: Compute additive attention weights over BiLSTM outputs and form the context vector $\mathbf{c}$ (Equations 10–12).
12: **Fully Connected Layers:**
13: Apply Dense layers (128 and 64 neurons, ReLU activation) with Dropout (rate 0.3) on $\mathbf{c}$.
14: **Output Layer:**
15: Apply Dense softmax layer for classification.
16: **Model Compilation:**
17: Use Adam optimizer (learning rate 1×10^{-3}) and sparse categorical cross-entropy loss.
18: **Callbacks:**
19: Implement Early Stopping (patience = 5, monitoring validation loss) and Learning Rate Reduction (factor 0.5, patience 3).
20: **Training and Evaluation:**
21: Train model on (X, Y) for a maximum of 50 epochs with batch size 64 and evaluate performance.
22: **return** Trained model

- Accuracy: Measures overall prediction correctness:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (16)$$

- Precision: Evaluates how many predicted malicious URLs are actually malicious:

$$Precision = \frac{TP}{TP + FP}. \quad (17)$$

- Recall (Sensitivity): Measures the ability to detect actual malicious URLs:

$$Recall = \frac{TP}{TP + FN}. \quad (18)$$

- F1-score: Provides a balance between precision and recall:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \quad (19)$$

3.9 Training and Validation Performance

Model performance is visualized using:

- Loss Curves: Show training and validation loss trends.
- Accuracy Curves: Indicate accuracy progression during training.

4 Result and Discussion

4.1 System Specifications

The system was set up with an Intel Dual-Core processor running at 2.20 GHz , 16GB of RAM , and a 500GB SSD , which ensures efficient processing, memory management, and storage for the handling of large datasets. Moreover, a 5.1 SVGA or VGA display is also utilized for the support of visualization and debugging phases, which are highly important for the development of deep learning models and their analysis. The proposed system was developed using Windows 11 as the operating system and Python as the programming language. The system utilizes deep learning libraries such as TensorFlow and PyTorch. The system uses Visual Studio as the IDE for coding and testing, and Jupyter Notebook for data preprocessing, model training, and visualization.

4.2 Exploratory Data Analysis

Using statistical measures and data representation, exploratory data analysis helps identify trends, validate hypotheses, and summarize the data. It also helps to gain a better understanding of the quality and nature of the data by using different data visualization tools, thus creating better knowledge of the data as a whole.

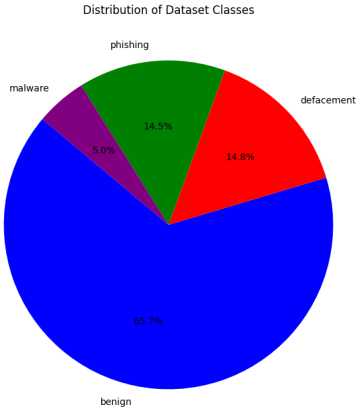

Figure 2: Pie chart showing the proportion of each class in the merged, QR-decoded ISCX URL 2016 dataset prior to any resampling.

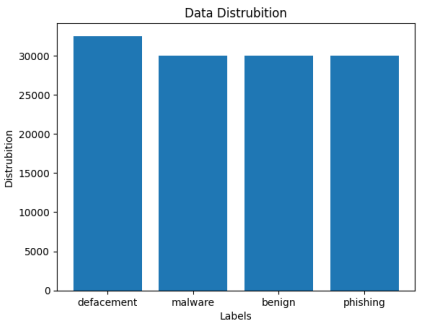

Figure 3: Balanced class distribution obtained after applying class-weighted resampling to the training split, used to mitigate the benign-class dominance shown in Figure 2.

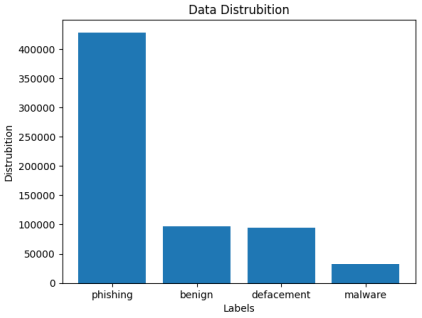

Figure 4: Imbalanced class distribution with phishing dominating – shown here as a counter-example illustrating the alternative, phishing-heavy imbalance pattern that the resampling strategy in Figure 3 is also designed to guard against.

The analysis consists of three graphs illustrating the class distributions in a dataset that is very important for the performance of machine learning models. The

first graph indicates that benign URLs are the majority with a percentage of 65.7%, which is a concern for biased model training. To handle this issue, the following approaches are proposed: oversampling, undersampling, and the use of class-weighted loss functions. The second graph shows a close to balanced distribution, which promotes the use of resampling methods to improve generalization. The third graph highlights a large class imbalance, especially in phishing URLs. This might cause incorrect classifications; solutions such as data augmentation and cost-sensitive learning can be applied.

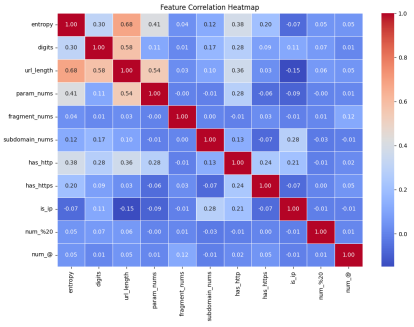

Figure 5: Correlation analysis of engineered lexical/statistical features derived from the QR-decoded URL strings (Section 3.4.2): entropy, digit count, URL length, subdomain count, parameter count, HTTPS usage, and IP-address presence.

Figure 5 shows the correlation between numerical features of URLs that are important for malicious URL detection. Significant positive correlations were found between entropy and URL length (0.68), and between the number of digits and URL length (0.58), which are related to the characteristics of URLs used in phishing attacks. A moderate correlation of 0.54 was found between URL length and the number of parameters, which might be used for malicious URL redirection. A slight correlation of 0.28 was found between the number of subdomains and "http," showing that URLs with a larger number of subdomains might use "http," which might not be a trusted protocol. The number of parameters was found to have a correlation of 0.28 with "http," which might be used for tracking and/or phishing attacks. In addition, a weak negative correlation was found between the presence of IP addresses and URL length, showing that URLs with IP addresses tend to be shorter (-0.15). A near zero correlation was found for percent-encoded characters and "@" presence, showing that these features are not important for classification. These findings point to the importance of entropy, digits, URL length, and parameter number in the classification of URLs, indicating that non-important features can be removed using PCA or RFE to improve classification accuracy in a Residual CNN (ResNet) model.

4.3 Proposed Model: ResCNN-BiLSTM with Attention Mechanism

Class	Precision	Recall	F1-Score
Benign	0.84	0.76	0.80
Malware	0.85	0.94	0.89
Defacement	0.92	0.88	0.90
Phishing	0.73	0.75	0.74

Table 3: Classification Report: Precision, Recall, and F1-Score for Each Class

The classification report assesses the performance of the model based on four classes: Benign, Malware, Defacement, and Phishing. The precision, recall, and F1-score are the metrics used to measure the performance of the model for each of these classes. For the Benign class, the precision is 0.84, recall is 0.76, and the F1-score is 0.80. For the Malware class, the precision is high at 0.85, recall is 0.94, and the F1-score is 0.89. For the Defacement class, the precision is the highest at 0.92, recall is 0.88, and the F1-score is 0.90. For the Phishing class, the precision is the lowest at 0.73, recall is 0.75, and the F1-score is 0.74. It can be seen that the performance of the model is satisfactory overall but has room for improvement for the Benign and Phishing classes.

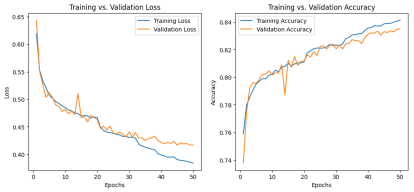

Figure 6: History of the performance of the model over 50 epochs, showing training/validation loss (left) and training/validation accuracy (right); the close tracking between the two curves supports the absence of substantial overfitting under the regularization settings in Table 2.

The training and validation accuracy curves demonstrate that the model is learning well for 50 epochs. The loss curve (left graph) demonstrates a steady drop in both the training and validation loss, which suggests that the model is working to reduce errors. Although the validation loss has some slight oscillations, it remains stable without a large difference between the training and validation loss, which suggests that there is not much overfitting. The accuracy curve (right graph)

demonstrates a steady increase in accuracy, with the training accuracy reaching about 84% and the validation accuracy following suit at about 83%. The slight oscillations in the validation accuracy are expected and do not suggest much instability.

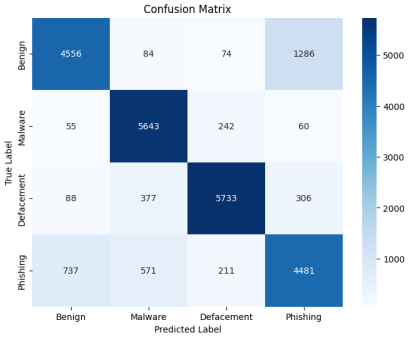

Figure 7: Confusion matrix for the four-class test-set predictions (Benign, Malware, Defacement, Phishing); rows are true labels and columns are predicted labels, with diagonal cells denoting correct classifications.

Figure 7 detailed the performance analysis of the classification models, where the model's ability to differentiate between the four classes: Benign, Malware, Defacement, and Phishing, is demonstrated. Each row of the confusion matrix corresponds to actual classes, and each column corresponds to the predicted classes. The diagonal values of the confusion matrix represent the correctly classified samples, and the off-diagonal values represent misclassifications.

For Benign URLs, the model was able to correctly classify 4,556 samples, but there was a misclassification of 1,286 Benign URLs into Phishing, which was the major confusion for the model. There were some misclassifications of Benign URLs into Malware (84 samples) and Defacement (74 samples), showing that the model was not able to differentiate between Benign URLs and the other classes of URLs.

For Malware URLs, the model was highly accurate, with 5,643 samples classified correctly, but there was a misclassification of 242 Malware URLs into Defacement, and 60 samples were misclassified as Phishing, showing that the model was highly effective in classifying Malware URLs, but there were some instances where the model was not able to differentiate between Malware and Defacement URLs, as these two classes might have some common features. In the case of Defacement URLs, the model was able to classify 5,733 instances correctly but incorrectly classified 377 instances as Malware and 306 instances as Phishing. A smaller number of 88 instances were incorrectly labeled as Benign. The incorrect classification between Defacement and Malware show a possible similarity between the two in their feature space.

For Phishing URLs, the model correctly classified 4,481 data points but incorrectly classified 737 as Benign, 571 as Malware, and 211 as Defacement. The large number of Phishing URLs classified as Benign implies that some Phishing URLs share similarities with Benign URLs, which may result in classification errors. Overall, it can be seen that the model has high classification performance, especially for Malware and Defacement classes. However, there are some incorrect classifications, especially between Benign and Phishing, and Malware and Defacement classes, which implies that the feature selection process can be improved or that more sophisticated methods, such as Residual CNN with RNN, can be used to improve classification performance.

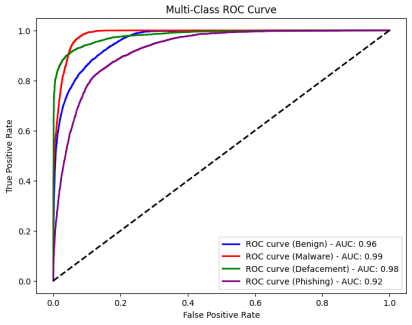

Figure 8: Multiclass ROC-AUC curve for each of the four classes, computed in a one-vs-rest fashion on the held-out test split.

From the multi-class ROC curve, it was observed that there was a strong correlation with precision, recall, and F1-score, validating the model's classification ability. The Malware class had the maximum AUC value of 0.99, with a strong F1-score and recall of 0.89 and 0.94, respectively, validating the model's ability to detect malware with a very high degree of accuracy. The Defacement class had a strong AUC value of 0.98, with a strong F1-score and precision of 0.90 and 0.92, respectively. The Benign class had a strong AUC value of 0.96, with lower recall (0.76) and precision (0.84), validating that some of the Benign URLs were misclassified into other classes. The Phishing class had the least AUC value of 0.92, with the least precision and recall of 0.73 and 0.75, validating that there was a problem in correctly classifying Phishing URLs and a high degree of misclassification was present for Phishing URLs.

4.4 Reconciling the Phishing AUC and F1-Score

The AUC-ROC measure considers the ranking effectiveness over all possible decision thresholds and is

computed in relation to the rest of the classes taken together, hence being able to remain high despite the considerable misclassification of a certain class that occurs due to a specific operating threshold. However, the F1 score of 0.74 is an indicator of how well the model performs in terms of the one specific threshold used during classification and is more affected by the confusion in the experiment between Phishing and Benign URLs. As shown in Figure 7, due to the use of a lower operating threshold in this study, the resulting lower F1-score corresponds to one of two possibilities for misclassification: (a) a significant proportion of phishing URLs, which are camouflaged as QR codes, get accepted, or (b) a significant number of legitimate QR code URLs are falsely classified as phishing URLs. The specific ratio of the former to the latter depends on the threshold settings. For a QR code scanner application that is used by end users, the former error case is much more critical from a safety standpoint, as the acceptance of a phishing URL will lead to a direct risk of stealing credentials, while the latter will simply be less convenient for the user. This gives rise to the proposed path forward for future research (Section 5).

4.5 Comparison with other Studies

Table 4: Performance Comparison with Models across different other studies

Model	Class	Precision	Recall	F1-Score
Light Weight DL [38]	benign	0.95	0.99	0.92
	Phishing	0.95	0.96	0.92
	Malware	0.97	0.94	0.94
CNN + BBOA [14] (overall, binary phishing task)	overall	0.92	–	–
Proposed Model	benign	0.84	0.76	0.80
	Phishing	0.73	0.75	0.74
	Malware	0.85	0.94	0.89
	Defacement	0.92	0.88	0.90

Significant variations can be seen in the classification accuracy of different categories of cyber threats using the proposed model and the Lightweight Deep Learning (DL) model. The proposed model shows lesser accuracy of 0.84 and recall of 0.76, which results in an F1-score of 0.80, while the Lightweight DL model excels in classifying benign occurrences with precision of 0.95, recall of 0.99, and an F1-score of 0.92. The Lightweight DL model again shows better performance than the proposed model, with precision of 0.95 and recall of 0.96, while classifying phishing attacks, while the proposed model shows precision of 0.73 and recall of 0.75, and an F1-score of 0.74. Both models show good performance while classifying malware threats, where the proposed model shows an accuracy of 0.89, while the Lightweight DL model shows an F1-score of 0.94. The proposed model shows good performance with an F1-score of 0.90 while detecting defacement attacks with an accuracy of 0.92 and recall of 0.88. The CNN+BBOA framework of [14], evaluated on a different (binary, Kaggle-sourced) phishing task, reported an overall precision of 0.92, which is broadly comparable to the Defacement-class precision achieved here (0.92) but higher than the Phishing-class precision of the proposed model (0.73); this gap is consistent with the four-class formulation used in this study being a strictly harder task than the binary phishing/benign setting in [14], and reinforces that direct numerical comparison across studies should be read alongside differences in dataset, class count, and task framing rather than as like-for-like ranking. The strong differentiation power of the proposed model is shown through the validation of the Receiver Operating Characteristic (ROC) curve, which shows high AUC values for all categories. In general, the proposed model's strong detection capability, especially in the defacement category, implies that the proposed model can be improved by optimizing additional feature selection and training approaches, although the Lightweight DL model has a better performance in known attacks.

5 Conclusion

The study was successful in developing a high-performance malicious URL detection system using the ResCNN-BiLSTM model, evaluated on URLs recovered from QR-code encoding/decoding (Section 3.2), which achieved high classification accuracy in distinguishing between malicious, defacement, phishing, and malware-related URLs [1, 46]. The proposed model, which utilizes optimized feature selection and extraction, outperforms conventional CNN and LSTM-based models, with ROC-AUC values up to 0.99 for malware detection, although it faced challenges in phishing classification, which still requires more refinement [21, 2], as discussed in Section 4.3. The study makes a significant contribution to the existing body of knowledge in the field of cybersecurity by advancing feature optimization methodologies, developing a strong multi-class threat detection model, and overcoming conventional binary classification limitations, with comprehensive performance benchmarking using AUC, precision, recall, and F1-score evaluation metrics [16, 8]. The study's contribution and limitations relative to the wider literature are further summarized in Table 1. Future work recommendations include improving phishing detection using more behavioral and lexical features, cost-sensitive threshold calibration for the Phishing class (Section 4.3), investigating ensemble learning using RNN or Transformer models, incorporat-

ing effective dataset augmentation to handle class imbalance problems, extending the current QR-to-text pipeline with a complementary image-based (Conv2D) branch that also inspects the visual QR-code matrix for tampering or spoofing, broadening the external benchmark comparison to additional multi-class malicious-URL datasets and baselines (Section 4.4), and applying the system in real-time settings with continuous learning capabilities to better adapt to new cyber threats [39, 3, 28].

References

[1] Ajlouni, N., Özyavaş, A., Takaoğlu, M., Takaoğlu, F., and Ajlouni, F. Medical image diagnosis based on adaptive hybrid quantum cnn. *BMC Medical Imaging*, 23(1):126, 2023.

[2] Al-Tarawneh, M., Al-irr, O., Al-Maaitah, K., Kanj, H., and Aly, W. Enhancing fake news detection with word embedding: A machine learning and deep learning approach, 07 2024.

[3] Aldakheel, E. A., Zakariah, M., Gashgari, G. A., Almarshad, F. A., and Alzahrani, A. I. A deep learning-based innovative technique for phishing detection in modern security with uniform resource locators. *Sensors*, 23(9):4403, 2023.

[4] Asadizanjani, N., Xi, C., and Tehranipoor, M. Chapter 8 - tracking and tracing methods for hardware assurance. In Asadizanjani, N., Xi, C., and Tehranipoor, M., editors, *Materials for Electronics Security and Assurance*, pages 119–128. Elsevier, 2024.

[5] Awasthi, D., Khare, P., and Kumar Srivastava, V. Rfdb: Robust watermarking scheme with fuzzy-dncnn using blockchain technique for identity verification. *Expert Systems with Applications*, 255:124554, 2024.

[6] Bhardwaj, A., Bharany, S., Abulfaraj, A. W., Osman Ibrahim, A., and Nagmeldin, W. Fortifying home iot security: A framework for comprehensive examination of vulnerabilities and intrusion detection strategies for smart cities. *Egyptian Informatics Journal*, 25:100443, 2024.

[7] Bouarroudj, R., Souami, F., Zohra Bellala, F., and Zerrouki, N. A reversible fragile watermarking technique using fourier transform and fibonacci q-matrix for medical image authentication. *Biomedical Signal Processing and Control*, 92:105967, 2024.

[8] Chow, Y.-W., Susilo, W., Wang, J., Buckland, R., Baek, J., Kim, J., and Li, N. Utilizing qr codes to verify the visual fidelity of image datasets for machine learning. *Journal of Network and Computer Applications*, 173:102834, 2021.

[9] Crossland, V., Dellwo, C., Bashar, G., and Dagher, G. G. Janus: Toward preventing counterfeits in supply chains utilizing a multi-quorum blockchain. *Blockchain: Research and Applications*, 4(4):100157, 2023.

[10] Devalla, V., Srinivasa Raghavan, S., Maste, S., Kotian, J. D., and Annapurna, D. D. murli: A tool for detection of malicious urls and injection attacks. *Procedia Computer Science*, 215:662–676, 2022. 4th International Conference on Innovative Data Communication Technology and Application.

[11] Do, N. Q., Selamat, A., Fujita, H., and Krejcar, O. An integrated model based on deep learning classifiers and pre-trained transformer for phishing url detection. *Future Generation Computer Systems*, 161:269–285, 2024.

[12] Do, N. Q., Selamat, A., Krejcar, O., and Fujita, H. Detection of malicious urls using temporal convolutional network and multi-head self-attention mechanism. *Applied Soft Computing*, 169:112540, 2025.

[13] Gupta, B. B., Gaurav, A., Arya, V., Attar, R. W., Bansal, S., Alhomoud, A., and Chui, K. T. Advanced bert and cnn-based computational model for phishing detection in enterprise systems. *CMES - Computer Modeling in Engineering and Sciences*, 141(3):2165–2183, 2024.

[14] Gupta, B. B., Gaurav, A., Attar, R. W., Arya, V., Bansal, S., Alhomoud, A., and Chui, K. T. A hybrid cnn-brown-bear optimization framework for enhanced detection of url phishing attacks. *Computers, Materials and Continua*, 81(3):4853–4874, 2024.

[15] Halbouni, A., Gunawan, T. S., Habaebi, M. H., Halbouni, M., Kartiwi, M., and Ahmad, R. Cnn-lstm: hybrid deep neural network for network intrusion detection system. *IEEE Access*, 10:99837–99849, 2022.

[16] Hasan, H. R., Musamih, A., Salah, K., Jayaraman, R., Omar, M., Arshad, J., and Boscovic, D. Smart agriculture assurance: Iot and blockchain

for trusted sustainable produce. *Computers and Electronics in Agriculture*, 224:109184, 2024.

[17] Hassan, A., Nizam-Uddin, N., Quddus, A., Hassan, S. R., Rehman, A. U., and Bharany, S. Navigating iot security: Insights into architecture, key security features, attacks, current challenges and ai-driven solutions shaping the future of connectivity. *Computers, Materials and Continua*, 81(3):3499–3559, 2024.

[18] Hilal, A. M., Abdalla Hashim, A. H., Mohamed, H. G., Nour, M. K., Asiri, M. M., Al-Sharafi, A. M., Othman, M., and Motwakel, A. Malicious url classification using artificial fish swarm optimization and deep learning. *Computers, Materials and Continua*, 74(1):607–621, 2022.

[19] Hong, J., Kim, T., Liu, J., Park, N., and Kim, S.-W. Phishing url detection with lexical features and blacklisted domains. *Adaptive autonomous secure cyber systems*, pages 253–267, 2020.

[20] Kim, S., Kim, J., and Kang, B. B. Malicious url protection based on attackers' habitual behavioral analysis. *Computers & Security*, 77:790–806, 2018.

[21] Li, K., Ma, W., Duan, H., Xie, H., and ZHU, J. Few-shot iot attack detection based on rfp-cnn and adversarial unsupervised domain-adaptive regularization. *Computers & Security*, 121:102856, 2022.

[22] Li, T., Kou, G., and Peng, Y. Improving malicious urls detection via feature engineering: Linear and nonlinear space transformation methods. *Information Systems*, 91:101494, 2020.

[23] Liu, R., Wang, Y., Xu, H., Qin, Z., Zhang, F., Liu, Y., and Cao, Z. Pmanet: Malicious url detection via post-trained language model guided multi-level feature attention network. *Information Fusion*, 113:102638, 2025.

[24] Liu, Y., Hu, Y., Lv, X., Zhou, S., Li, J., and Liu, S. A blockchain and ipfs-aided anonymous traitor tracing scheme based on puncturable encryption in industrial internet of things. *Computers and Electrical Engineering*, 122:109896, 2025.

[25] M., G. and Sethuraman, S. C. A comprehensive survey on deep learning based malware detection techniques. *Computer Science Review*, 47:100529, 2023.

[26] Mendonca, R. V., Silva, J. C., Rosa, R. L., Saadi, M., Rodriguez, D. Z., and Farouk, A. A lightweight intelligent intrusion detection system for industrial internet of things using deep learning algorithms. *Expert Systems*, 39(5):e12917, 2022.

[27] Mondal, D. K., Singh, B. C., Hu, H., Biswas, S., Alom, Z., and Azim, M. A. Seizemaliciousurl: A novel learning approach to detect malicious urls. *Journal of Information Security and Applications*, 62:102967, 2021.

[28] Naik, R. L., Jain, D. S., and Bairam, D. M. A heuristic assisted cyber attack detection system using multi-scale and attention-based adaptive hybrid network. *Journal of Information Security and Applications*, 89:103970, 2025.

[29] Okey, D. O., Dadkhah, S., Molyneaux, H., Rodríguez, D. Z., and Kleinschmidt, J. H. ipaseciot: An intelligent pipeline for automatic and adaptive feature extraction for secure iot device identification and intrusion detection. *Internet of Things*, page 101802, 2025.

[30] Okey, O. D., Maidin, S. S., Adasme, P., Lopes Rosa, R., Saadi, M., Carrillo Melgarejo, D., and Zegarra Rodríguez, D. Boostedenml: Efficient technique for detecting cyberattacks in iot systems using boosted ensemble machine learning. *Sensors*, 22(19):7409, 2022.

[31] Okey, O. D., Rodríguez, D. Z., Guimarães, F. G., and Kleinschmidt, J. H. Cafiks: Communication-aware federated ids with knowledge sharing for secure iot connectivity. *IEEE Access*, 2025.

[32] OYEDEMI, O., Rodriguez, D. Z., Rosa, R. L., and Ugochukwu, O. M. Containerization approach for secure internet of medical things (iomt) communication protocols. *INFOCOMP Journal of Computer Science*, 23(2), 2024.

[33] Oyedemi, O. A., Rosa, R. L., Matthew, U. O., Ogundele, L. A., Ogunwale, Y. E., and Rodríguez, D. Z. Privacy-preserving federated data governance framework for secure parameter exchange in distance education. In *2025 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–6. IEEE, 2025.

[34] Patgiri, R., Biswas, A., and Nayak, S. deepbf: Malicious url detection using learned bloom filter

and evolutionary deep learning. *Computer Communications*, 200:30–41, 2023.

[35] Rasheed, B., Khan, A., Kazmi, S. M. A., Hussain, R., Piran, M. J., and Suh, D. Y. Adversarial attacks on featureless deep learning malicious urls detection. *Computers, Materials and Continua*, 68(1):921–939, 2021.

[36] Rashid, F., Ranaweera, N., Doyle, B., and Seneviratne, S. Llms are one-shot url classifiers and explainers. *Computer Networks*, 258:111004, 2025.

[37] Rathod, T., Jadav, N. K., Tanwar, S., Alabdulatif, A., Garg, D., and Singh, A. A comprehensive survey on social engineering attacks, countermeasures, case study, and research challenges. *Information Processing & Management*, 62(1):103928, 2025.

[38] Sarkhi, M. and Mishra, S. Detection of qr code-based cyberattacks using a lightweight deep learning model. *Engineering, Technology & Applied Science Research*, 14:15209–15216, 08 2024.

[39] Serrano, W. Cyberaibot: Artificial intelligence in an intrusion detection system for cybersecurity in the iot. *Future Generation Computer Systems*, 166:107543, 2025.

[40] Sheikhi, S. and Kostakos, P. Safeguarding cyberspace: Enhancing malicious website detection with psooptimized xgboost and firefly-based feature selection. *Computers & Security*, 142:103885, 2024.

[41] Song, J., Gao, K., Shen, X., Qi, X., Liu, R., and Choo, K.-K. R. Qrfence: A flexible and scalable qr link security detection framework for android devices. *Future Generation Computer Systems*, 88:663–674, 2018.

[42] Soon, J. M. and Manning, L. Developing anti-counterfeiting measures: The role of smart packaging. *Food Research International*, 123:135–143, 2019.

[43] Ugochukwu, O. M., Rosa, R. L., Adenike, O. O., and Rodriguez, D. Z. Advancing cybersecurity use of sensitive data in electronic healthcare system: A review of privacy and regulations. *INFOCOMP Journal of Computer Science*, 23(2), 2024.

[44] Wu, B., Zou, F., Zhang, C., Yu, T., and Li, Y. Multi-field relation mining for malicious http traffic detection based on attention and cross network. *Journal of Information Security and Applications*, 73:103411, 2023.

[45] Yang, J. and Li, J. Application of deep convolution neural network. In *2017 14th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 229–232. IEEE, 2017.

[46] Yang, Y., Du, Y., Gupta, V. K., Ahmad, F., Amiri, H., Pan, J., Aghbashlo, M., Tabatabaei, M., and Rajaei, A. Exploring blockchain and artificial intelligence in intelligent packaging to combat food fraud: A comprehensive review. *Food Packaging and Shelf Life*, 43:101287, 2024.

[47] Yasin, A., Fatima, R., JiangBin, Z., Afzal, W., and Raza, S. Can serious gaming tactics bolster spear-phishing and phishing resilience? : Securing the human hacking in information security. *Information and Software Technology*, 170:107426, 2024.

[48] Zegarra Rodriguez, D., Daniel Okey, O., Maidin, S. S., Umoren Udo, E., and Kleinschmidt, J. H. Attentive transformer deep learning algorithm for intrusion detection on iot systems using automatic xplainable feature selection. *Plos one*, 18(10):e0286652, 2023.