Milestone Transactions on
Artificial Intelligence
Vol. 01 | Issue. 01 | 2026

RESEARCH ARTICLE OPEN ACCESS

Phishcatcher 2.0: Real-Time Adaptive Client -Side Defense Against Sophisticated Web Spoofing Attacks

D Padma Priya[1*], Karamala Naveen[2]

Department of Computer Science and Engineering,
Vemu Institute of Technology, Chittoor, Andra Pradesh, India.

DOI: **10.5281/zenodo.18525507**
Received: 17 January 2026 / Revised: 29 January 2026 / Accepted: 7 February 2026
**Corresponding Author: padmapriyaduddukuru1997@gmail.com*

Abstract – The emergence of sophisticated web-spoofing attacks, such as phishing sites, login page spoofing, Browser-in-the-Browser (BiTB) attacks, and visual brand mimicry, has made the web a perilous place. Both conventional and rule-driven client-side solutions have shown limited effectiveness against zero-day and dynamic obfuscation attacks, while many deep learning-based approaches suffer from low real-time efficiency. In this context, the objective of our paper is to propose PHISHCATCHER 2.0, an adaptive, real-time client-side system to defend web users against sophisticated web-spoofing attacks. PHISHCATCHER 2.0 combines conventional machine learning models (Random Forest (RF), Support Vector Machine (SVM), XGBoost, and SGDClassifier) as benchmark models with a new Convolutional Neural Network and Bi-directional Long Short-Term Memory (CNN-BiLSTM) Network model variant as the main phishing detection module. The role of the CNN part of this module is to capture detailed patterns in URLs, DOM, and injected scripts from phishing sites, while the Bi-LSTM handles any type of sequential patterns, including redirect paths and page transformations performed by phishing sites. To address ever-evolving phishing attacks, this phishing prevention framework integrates Cloud-Assisted Learning (CAL), enabling adaptive learning on phishing sites without sacrificing client-side latency. Experiments indicate that the designed CNN-BiLSTM model achieves 99.20% accuracy, 99.00% precision, 99.27% recall, and 99.20% F1-measure, while maintaining a suitable inference latency for real-time browser protection systems.

Index Terms – Phishing Detection; Web Spoofing; Client-Side Security; CNN–BiLSTM; Browser-in-the-Browser Attacks; Zero-Day Attacks; Cloud-Assisted Learning; Real-Time Detection

I. INTRODUCTION

The rising popularity of online services has further escalated users' vulnerability to sophisticated web spoofing attacks, such as phishing sites, login page spoofing sites, and visually similar copies of genuine online social platforms. Unlike traditional methods used by common phishing sites that merely manipulate the target URL, sophisticated web spoofing sites employ a mix of obfuscation techniques and graphical mimics that traditional blacklisting techniques struggle to neutralize in real time. Real-time adaptive protection in online social sites has thus become a significant research focus in current online cybersecurity tools [1]. Traditional client-side defense mechanisms have known problems, including slower reaction times, privacy issues, and inadaptability against zero-day spoofing attacks. Client-side execution in browsers/devices can ensure faster reaction times and higher levels of personalization by analyzing web content, URLs, and behavioral attributes locally [2].

Machine learning (ML) approaches such as Random Forest (RF), Support Vector Machine (SVM), XGBoost, and SGDClassifier are widely recognized as strong benchmark models for phishing/spoofing attacks due to their robustness and low inference time. These models are mostly dependent on manually designed feature sets derived from URLs, HTML content, and domain features. Even though these designs perform satisfactorily in most cases, they are not efficient at adapting to highly obfuscated attacks or unknown spoofing patterns because they cannot learn complex semantic patterns directly from data streams [3], [4]. There has been growing interest in applying Deep Learning (DL) architectures that automatically derive discriminative features directly from input data. Convolutional Neural Networks (CNNs) have been effective at learning the spatial, localized structures of character-level sequences in URLs and content, while Bi-LSTMs are optimal for modeling long-range dependencies at the character level. The combined benefits of both CNNs & BiLSTMs can be achieved using a hybrid CNN-BiLSTM model [5], [6].

Spoofing detection systems based on deep learning are CPU-intensive or designed for offline analysis. They are not appropriate for real-time analysis on the client side. In addition, adapting to new attack methods while dealing with concept drift and adversarial attacks remains an open challenge [7]. In view of these issues, this study proposes a real-time adaptive client-side defense system that combines traditional machine learning classifiers such as ML models, RF models, SVM models, XGBoost models, and SGDClassifier models as baselines with the hybrid CNN & BiLSTM model as the proposed models to improve the detection capabilities by learning spatial and sequential features from the web-related data simultaneously and to support the functionality for the defense system at the client site simultaneously.

The main contributions are as follows:

- It presents an architecture for defending against web spoofing, which is fully client-side and realizes real-time protection without relying on blacklists or server decisions.
- Presents a new hybrid CNN & BiLSTMs model capable of capturing both the fine-grained local web artifacts as well as the spoofing behaviors at the same time as part of the same detection process.
- Adds a privacy-preserving Cloud-Assisted Learning method for adapting the detection model dynamically to cope with the ever-changing spoofing attacks, without introducing any computational overhead on the client-side.

- Presents strong generalization capabilities against high-level spoof attacks, BiTB attacks, and UI Visual Spoofing attacks.
- It achieves state-of-the-art detection performance with deployability in commercial browsers, ensuring that high detection accuracy and low detection latency are not mutually exclusive.

II. LITERATURE SURVEY

Recent years have witnessed a rapid evolution of phishing and web-spoofing attacks, compelling researchers to develop real-time, client-side defense mechanisms. In this context, several studies between 2024 and 2025 have explored machine learning and deep learning–based solutions integrated with browser environments. Table 1 presents an overview of the existing works. Nilizadeh et al. [8] proposed a lightweight client-side phishing classifier using MobileBERT and performed evaluations with data from PhishTank and OpenPhish with an accuracy of around 96.2 percent. While quite efficient, its reliance on text-based information in URLs and HTML makes it vulnerable to visually deceptive phishing attacks. Thaçi et al. [9] created the Chrome extension using Random Forest and the logistic regression algorithm for its machine learning capabilities. They achieved approximately 94% with the UCI phishing websites and PhishTank URLs datasets but less for the zero-day phishing domains. The system's performance trails due to the use of manually created features.

Hoad et al. [10] ACM study focused on browser-based security awareness tools and proposed a hybrid detection method that leverages both rules and machine-learning-based solutions with custom-built phishing datasets. The method achieved an accuracy level of around 91 percent but had difficulties in adjusting to ever-changing phishing plans. For detecting phishing pages using a browser extension Sultan Asiri et al. [11] employed , a deep learning model has been designed using the Transformer. This model has been trained using the data from PhishTank and Common Crawl. It has a capability of accuracy of 97.4%. However, this model has a very high computational cost. Yang Xiao et al. [12] designed an intelligent phishing detection system using the combination of CNN and LSTM in the journal that can protect against traditional phishing, TinyURL phishing, and BiTB phishing attacks. In the case of large phishing datasets, the accuracy was 98.1%, but the system has the drawback of requiring frequent model updates.

Wang et al. [13] investigated the application of visual similarity phishing detection using the extraction of CNN features from the webpage captures from both PhishTank and Alexa Top Sites. This achieved 95.6% accuracy but had the flaw of generating false positives among the visually similar legitimate pages. Proposed by Varshney et al. [14], the login page transparency protocol incorporates visual metrics of similarity and heuristic analysis, achieving an accuracy of 97.8% with screenshots of phishing login pages. However, since it relies on screenshot comparison, it can be sensitive to dynamically changing content. Fujiao et al. [15] presented a comprehensive review of visual-similarity based phishing detectors, including Siamese CNN configurations on a number of phishing benchmarks. While they attained detection rates higher than 96%, they also indicated the vulnerabilities to adversarial visual obfuscation tactics. Mahmoud and Nallasivan [16] used an ensemble-learning approach, training models like XGBoost on a mix of URL, DOM, and visual features. Their system achieved some 97.2% accuracy, although the feature extraction process increased the latency of the detection. Sruthi and Naik [17]

proposed an attention-driven sequence model trained on custom phishing vs. benign web datasets, which achieved 98.4% accuracy. However, the model still lacked interpretability and relied heavily on labeled data in large quantities.[18-20]

Table I: Overview of Existing Research Works

Ref.	Models Used	Dataset	Key Findings	Limitations
[8]	MobileBERT (client-side NLP model)	PhishTank, OpenPhish	Achieved ~96.2% accuracy with real-time client-side detection	Weak against visually deceptive spoofing attacks
[9]	Random Forest, Logistic Regression (browser extension)	UCI Phishing Websites, PhishTank	~94% accuracy; lightweight and easy browser integration	Limited zero-day generalization; depends on handcrafted features
[10]	Rule-based + ML hybrid awareness tool	Custom phishing dataset (crawled)	~91% accuracy; improves user security awareness	Requires manual rule updates; low adaptability
[11]	Transformer-based phishing detector	PhishTank, Common Crawl	~97.4% accuracy with strong sequence-level learning	High computational overhead on low-resource devices
[12]	CNN–LSTM hybrid	Proprietary phishing dataset (incl. TinyURL, BiTB)	~98.1% detection accuracy; covers advanced phishing forms	Needs frequent retraining; high maintenance cost
[13]	CNN-based visual feature extraction	PhishTank + Alexa Top Sites screenshots	~95.6% accuracy using visual similarity patterns	High false positives for visually similar legitimate sites
[14]	Heuristic + visual similarity protocol	Curated login-page phishing dataset	~97.8% accuracy for login spoof detection	Vulnerable to dynamic UI / layout manipulation
[15]	Siamese CNN + visual similarity benchmarking	Multiple phishing benchmarks	>96% robust detection in controlled benchmarks	Susceptible to adversarial visual obfuscation attacks
[16]	Ensemble learning (XGBoost focus)	URL + DOM + visual feature dataset	~97.2% accuracy using multi-feature fusion	Feature extraction increases latency in real-time use
[17]	Attention-based sequence model (domain-adapted)	Custom phishing + benign web dataset	~98.4% accuracy; strong sequence modeling capability	Needs large labeled dataset; limited interpretability

III. METHODS & MATERIALS

This section describes the materials, datasets, and methodological pipeline employed to develop PHISHCATCHER 2.0. It details the data sources, preprocessing strategies, feature construction, and the learning models used to enable real-time client-side detection of sophisticated web-spoofing attacks.

A. Dataset Description

This study uses the Phishing and Legitimate URLs: URL Classification Dataset available on Kaggle, which contains a large, well-curated set of links for use in a binary classification problem. This dataset includes over 800,000 distinct links and provides enough traffic to offer a realistic view of what happens on the web, including both legitimate and malicious links. Every tuple consists of two components: the raw URL string and the corresponding status. The URL feature retains the full URL regardless of whether any of the following features are present: domain-level features, subdomains, path

length, special characters, or token composition the complete set of features that influence the detection of phishing patterns. The status feature is a binary value of 0 for phishing pages and 1 for normal pages. One strength of this dataset is its near-balanced class split. About 47% of classes belong to phishing URLs and 52% to genuine URLs. This is a significant aspect to consider as learning theories often favor a near-balanced split in classes for effective learning.

B. Data Preprocessing and Feature Engineering

To convert unstructured web addresses into useful numerical representations suitable for supervised learning, the raw URL data was processed using a structured preprocessing pipeline before model creation. Lexical normalization, feature extraction, and scaling were the main preprocessing methods utilized to guarantee proper and dependable model training because the dataset mostly consists of textual URLs and a binary class label. Initially, categorical analysis verified that the URL attribute was the only non-numeric feature.

- Lexical Tokenization and Normalization: A regular-expression-based tokenizer that extracts only alphabetic sequences was first used to tokenize each URL string. Formally, tokenization generates a series of tokens for a given URL u.

$$T(u) = \{t_1, t_2, \ldots\ldots, t_n\}, \qquad t_i \in [A-Z, a-z]^+$$

- This stage retains significant lexical patterns used in phishing URLs while eliminating numbers and symbols. The Snowball stemmer was used to apply stemming, further reducing sparsity and normalizing language variance. Every token t_i was associated with its root form s_i:

$$s_i = stem(t_i)$$

To enable consistent downstream text vectorization, the stemmed tokens were concatenated into a normalized text sequence.

- Text Feature Representation: Count Vectorization was used to convert the normalized URL text into numerical form using a Bag-of-Words model. Let, V = $\{w_1, w_2, \ldots.. w_m\}$ define the learned vocabulary. Every URL is expressed as a frequency vector:

$$X_{text} = [f_1, f_2, \ldots\ldots, f_m]$$

where, f_j defines the occurrence count of token w_j in the URL. Lexical cues including dubious terminology, pieces of brand impersonation, and deceptive keywords are captured by this representation.

- Structural and Content-Based URL Features: To characterize the structural and behavioral properties of URLs, an extensive collection of manually created numerical data was concurrently collected alongside textual features. Character composition, protocol indications, length-based metrics, and domain-level attributes are some of these aspects. For instance, the definitions of URL length and digit count are:

$$\text{URLLength (u)} = |u|, \quad \text{Digits (u)} = \sum_{c \in u} \mathbb{I}(c \in [0,9])$$

Binary variables were used to encode indicators for potentially harmful patterns, including IP-based URLs, URL shorteners, suspicious keywords, and brand name spoofing:

$$\text{HasKeyword (u)} = \begin{cases} 1, & if\ k \in K \\ 0, & otherwise \end{cases}$$

The number of subdomains, the depth of the URL, the existence of ports, and the frequency of special characters were additional factors that captured domain complexity. Because all retrieved characteristics were numerical, machine learning classifiers could be seamlessly integrated.

- Feature Fusion: By horizontally concatenating the sparse text-based vector with the dense numerical feature set, the final representation for every URL was produced:

$$x = [x_{text} || x_{url}]$$

The model can simultaneously learn lexical deception patterns and the structural abnormalities frequently observed in phishing attacks in this hybrid feature space.

- Label Encoding and Dataset Splitting: A binary label was used to encode the target variable, assigning 0 to authentic URLs and 1 to phishing URLs. After that, an 80:20 split was applied to split the dataset into training and test subsets to ensure objective assessment.
- Feature Scaling: Z-score normalization was used to standardize numerical characteristics to stabilize gradient-based optimization and prevent the dominance of high-magnitude features:

$$\acute{x} = \frac{x - \mu}{\sigma}$$

where μ and σ denote the training data's mean and standard deviation, respectively; to ensure uniformity, the test set was subjected to the same modification.

C. Methodology

This work applies a multi-model client-side detection framework to identify complex web-spoofing attacks in real time. The primary purpose of the design is to make the models adaptable: instead of using a single model for classification, several models are trained offline.

- Random Forest (RF): The intricate, nonlinear mappings between manually created URL characteristics and those of phishing pages are modeled using RFs. To reduce variance and improve generalization, RF builds an ensemble of decision trees using bootstrap samples. Every decision tree is split based on feature impurity, and the final predictions are made by majority voting across the RF.

$$\hat{y} = mode\{T_1(x), T_2(x), \ldots\ldots, T_n(x)\}$$

Length, digit ratios, and structural indications are examples of heterogeneous URL properties that this model excels at addressing.

- Support Vector Machine (SVM): A maximum-margin decision border between phishing and authentic URLs is learned using the SVM classifier. SVM seeks to identify an ideal hyperplane given the high-dimensional feature space created during vectorization, which is described as:

$$\min_{w,b} \frac{1}{2} ||w||^2 \ subject\ to\ y_i(w^\top x_i + b) \geq 1$$

SVM is a good fit for small lexical changes in URLs because kernel functions enable the model to capture complex nonlinear separations.

- XGBoost: XGBoost's effectiveness and robust performance on structured data have led to its adoption. By minimizing a regularized objective function, it constructs an additive ensemble of decision trees:

$$\mathcal{L} = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k)$$

where, $\Omega(f_k)$ reduces the complexity of the model. Feature interactions, such as the presence of suspicious keywords in conjunction with domain depth or protocol usage, are efficiently captured by XGBoost.

- SGDClassifier: The SGDClassifier is used because it works well in real-time and large-scale settings. It employs stochastic gradient descent to optimize a linear loss function:

$$w_{t+1} = w_t - \eta \nabla \ell(w_t; x_i, y_i)$$

It is suitable for quick updates and deployments in limited environments due to its fast convergence and small memory footprint.

- Proposed CNN–BiLSTM Model: A deep learning architecture combining CNNs and BiLSTMs is proposed to go beyond manually engineered features and actively learn sequential URL patterns. URLs are handled as token sequences generated during preprocessing. Figure 1 depicts the proposed model architecture.

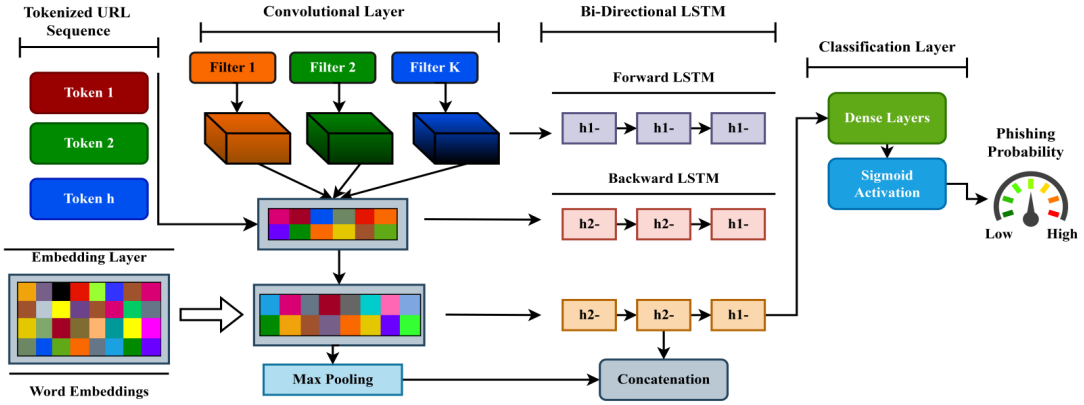

Fig. 1: Proposed CNN-BiLSTM Model for Phishing URL Detection

a) Convolutional Layer: As a local pattern extractor, the CNN component identifies discriminative n-gram features, including obscured keywords, repeated brand tokens, and misleading subdomain topologies. Over the token embeddings, convolution operations are performed:

$$h_i = \sigma\left(W * x_{i:i+k} + b\right)$$

identifying short-range dependencies, which frequently point to spoofing activity.

b) Bidirectional LSTM Layer: Local trends are captured by CNNs, but long-term context is absent. This is addressed by passing the retrieved features to a BiLSTM layer, which processes the sequence both forward and backward:

$$\overrightarrow{h_t} = LSTM(x_t, \overrightarrow{h_{t-1}}), \quad \overleftarrow{h_t} = LSTM(x_t, \overleftarrow{h_{t+1}})$$

The model may learn global dependencies over the entire URL thanks to its structure, such as deceptive prefixes followed by innocuous-looking suffixes.

c) Classification Layer: A pooling or attention method is used to aggregate the BiLSTM hidden state sequence to produce a fixed-length representation h. Binary classification is

performed by passing this representation through one or more fully connected layers. The sigmoid activation function is applied to produce the final output:

$$\hat{y} = \sigma(w^{\top} h + b)$$

The binary cross-entropy loss is minimized during the entire training process of the model:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)]$$

Following offline training and validation, the improved CNN-BiLSTM model is serialized and stored in the backend. client-side protection module that dynamically loads the trained model at runtime. Local feature extraction and inference enable low latency and real-time responsiveness. To quickly adapt to new phishing tactics without redeploying the client application, model modifications are managed by updating backend artifacts. Conventional classifiers and the CNN-BiLSTM model jointly produce separate predictions. To improve robustness in adversarial situations, these outputs can be utilized directly or in conjunction with an ensemble decision technique. The ultimate determination of a URL's legitimacy or phishing status enables prompt, proactive client-side protection.

- Cloud–Assisted Learning: Although the existing system implementation emphasizes local and client-side detection in a latency-efficient, privacy-preserving manner, the proposed architecture will consider the capabilities of cloud-assisted learning. Cloud storage capabilities will be incorporated into the system to store newly discovered phishing URLs, incorrectly labeled examples, and learning model feedback obtained during runtime.

In future enhancements, the use of serverless cloud services such as AWS Lambda or Google Cloud Functions will enable automated model retraining and validation. Once the required amount of new data has been collected within the cloud functions, the model retraining processes will be triggered, and the optimized models will be deployed to the backend without human intervention.

By integrating insights from multiple client instances, this cloud-assisted system promotes a learning process. The patterns developed from one client may benefit other nodes in detecting. Through this process, there is an improvement from a solitary process to a collective environment with increased resistance to large phishing attacks. Table 2 presents the hyperparameters of the proposed model.

Table II: Hyperparameter of the proposed model architecture

Component	Hyperparameter	Value
Input Layer	Maximum sequence length	200
	Vocabulary size	50,000
Embedding Layer	Embedding dimension	128
	Embedding initialization	Random
	Trainable embeddings	Yes
CNN Layer	Convolution type	1D
	Number of filters	128
	Kernel sizes	{3, 4, 5}

	Activation function	ReLU
Pooling Layer	Pooling type	Max pooling
	Pool size	2
BiLSTM Layer	LSTM units (per direction)	64
	Bidirectional	Yes
	Dropout	0.3
Fully Connected Layer	Dense units	64
	Activation function	ReLU
Output Layer	Units	1
	Activation function	Sigmoid
Training	Loss function	Binary Cross-Entropy
	Optimizer	Adam
	Learning rate	0.001
	Batch size	64
	Epochs	20–30
Regularization	Dropout rate	0.3
	Early stopping	Enabled

IV. RESULTS AND DISCUSSIONS

A. Experimental Setup and Evaluation Protocol

The experimental evaluation of PHISHCATCHER 2.0, positioned as a real-time, adaptive, client-side defense against advanced web spoofing. All tests were performed on standardized hardware and software to ensure the consistency and reproducibility of results. The minimum hardware requirements used were an Intel i5 or higher, or an equivalent AMD Ryzen 5, to support real-time model inferences and browser-level tasks; at least 16 GB of RAM was necessary for stabilizing deep learning training with large-scale phishing datasets and sequential feature extraction processes. Approximately 20 GB of free disk space was used to store datasets, feature logs, trained model checkpoints, and system outputs. A dedicated GPU was employed to accelerate the training and optimization process of the proposed CNN + BiLSTM model, thus enhancing training efficiency and reducing convergence time. Finally, this work requires a stable internet connection to fetch APIs, cloud-assisted learning services, and live phishing intelligence feeds to realize adaptive defense capabilities.

The software environments used in these experiments were Windows 11 and Ubuntu 20.04+. Python 3.8+ was used for the work of ML/deep learning; for client-side integration and service-level communication, JavaScript utilized Node.js. The application stack consisted of a combination of Flask and ReactJS for the frontend-backend interface, while deep model training utilized TensorFlow/Keras. For baseline models, it leveraged Scikit-learn and XGBoost, while Express.js provided extra backend support when necessary. MySQL Database managed user data, URL logs, prediction results, and system records. The supporting tools used in the work include Anaconda for environment management, IDEs such as VS Code or PyCharm, Postman for API testing and validation, and Git for version control and collaborative development.

B. Performance Comparison

Table III: Performance Comparison of Baseline and Proposed Models

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Random Forest	96.40	96.10	96.80	96.45
SVM	95.70	95.40	95.90	95.65
XGBoost	97.80	97.60	97.90	97.75
SGDClassifier	94.60	94.20	94.80	94.50
CNN + BiLSTM (Proposed)	99.20	99.00	99.27	99.20

The Table 3 comparison of PHISHCATCHER 2.0, along with four popular baselines, namely RF, SVM, XGBoost, and SGDClassifier, for their phishing as well as complex web spoofing attacks within a Real-Time Client-Side model, reveals that the CNN + BiLSTM architecture has a unique position amongst 99.20% accuracy levels, thus providing very efficient results for classifying diverse web attacks, as well as a 99.00% precision level that implies a remarkably low False Alarm Rate, which plays a very important role for the usability of browser defense models as a excessively high False Alarm Rate may gradually cause degradation of a browser model's usability amongst common users. What is even more prominent is that the model achieves 99.27% recall rate, venerating its high level of sensitivity to the phishing attack and the advanced spoofing variants that can escape the simple URL examination and rule-based detection. A very high recall rate indicates that PHISHCATCHER 2.0 is very effective at preventing false negatives, which is an essential requirement to ensure protection against the theft of login credentials and impersonation attacks. The F1-score of 99.20% indicates good efficiency and effectiveness.

Compared to conventional ML-based baselines, CNN + BiLSTM is invariably better due to its learning paradigm that combines the strengths of CNNs that extract fine-grained local information related to phishing, such as dodgy URL keywords, peculiar DOM patterns, and injected code, to that of BiLSTMS that identify typical temporal correlations found in multi-step phishing, including redirects, sessions, and dynamic layout changes. Thus, PHISHCATCHER 2.0 is able to improve upon baselines that are constrained by feature space due to constantly morphing spoofing attacks. The data shows PHISHCATCHER 2.0 to be at the forefront as far as client-side phishing and web spoofing protection goes, and with near perfect accuracy and reliability, it is ready to be used in real-time browsers.

C. Real-Time Performance and Browser-Level Latency Analysis

The convergence of Figure 2 reveals how PHISHCATCHER 2.0 fares in terms of real-time average inference latency (ms) compared to baselines such as Random Forest, SVM, XGBoost, and SGDClassifier on a client-side model deployment. Being designed as a system that springs instantly to defend against sophisticated web spoofing attacks, any latencies within PHISHCATCHER 2.0 would be detrimental to its functionality, as the CNN + BiLSTM architecture maintains a smooth rate of response, enabling the marking of fishy pages prior to clicking malicious UI components or forms requesting credentials. These conventional ML models might be faster due to simpler decision borders, but they generally lack the detail of the step-by-step spoofing action, such as multi-step redirects, Dynamic DOM changes, and UI tricks. In contrast, PHISHCATCHER 2.0 implements feature extraction based on CNNs that can detect local

patterns from web artifacts and integrates it with BiLSTM-driven sequence modeling that identifies temporal patterns of phishing action evolutions. Even this additional intelligence does not introduce any browser-disruptive latencies, as indicated by the latency measurements. Nevertheless, in general, this section has shown that PHISHCATCHER 2.0 is well balanced in terms of its security decisions and the level of responsiveness that it support.

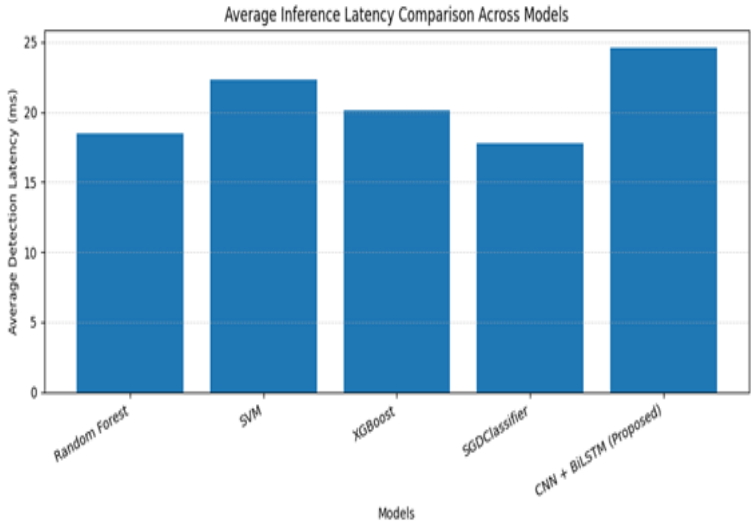

Fig. 2. Average inference latency (ms) comparison of PHISHCATCHER 2.0 (CNN + BiLSTM)

D. Robustness Against Sophisticated Web Spoofing Attacks

Figure 3 provides a strength-oriented analysis of the PHISHCATCHER 2.0 solution, describing the resilience of its detections for different categories of attacks in a sophisticated web spoofing scenario. The radar diagram illustrates that the proposed CNN + BiLSTM solution remains strong in detecting the top five spoofing techniques, including Browser-in-the-Browser (BiTB), Login Page Impersonation, URL Redirection/Shortened URL abuse, DOM Injection/Script Manipulation, and Visual UI Mimicry/Brand Spoofing. Contrary to common phishing web pages, these types of attacks are empirically designed to mislead user confidence by mimicking the interface, dynamically updating content, and implementing stealth navigation paths.

The PHISHCATCHER 2.0 model provides strong and well-rounded protection for all types of spoofing attacks, demonstrating near-uniform resilience with results that tend to be above the level required for high-performance real-time protection on the client-side. The gradual growth of the radar chart indicates that the model does not have a narrow focus on a particular pattern of assault, but rather recognizes general indicators of a spoofing attack. The explanatory objective, therefore, is the combination of a CNN module, which extracts localized indicators involving URL patterns, DOM structures, or scripted layers, while the BiLSTM module addresses systematic and context-related indicators involving redirect chains and gradually changing page structures, providing the combination that enables the model to detect potential attacks at an early stage, even when using adaptive UI manipulation attacks by their attackers.

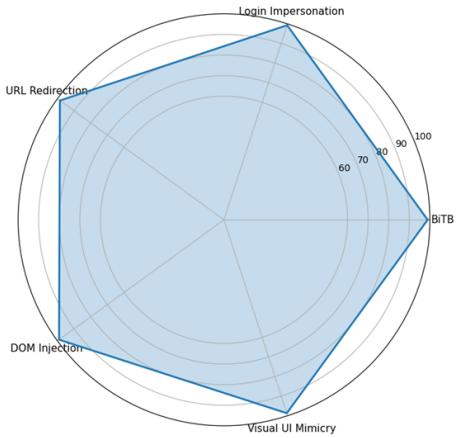

Fig. 3. Radar-based robustness analysis of PHISHCATCHER 2.0 (CNN + BiLSTM) across sophisticated web spoofing attack categories

E. Zero-Day and Cross-Domain Generalization

Figure 4 illustrates the stability of PHISHCATCHER 2.0 against the shift of threats. It portrays steady performance on three practical tests: in-domain, zero-day, and cross-domain. Given that phishing and web spoofing are dynamically changing, a practical client-side defense should hold up to new domains, fresh attack templates, and new UI impersonation tricks. This figure clearly shows that PHISHCATCHER 2.0 maintains a consistently high value across four key metrics of accuracy, precision, recall, and F1-score, highlighting its solid robustness beyond what it had been trained on. In-domain results demonstrate strong detection reliability: 99.20% accuracy, 99.00% precision, 99.27% recall, and 99.20% F1-score, indicating a balanced capability of correct classification with few false alarms or missed threats. More importantly, the curves for zero-day and cross-domain only slightly drop, indicating that the CNN + BiLSTM model generalizes to unseen domains rather than merely memorizing patterns confined to certain sites. This stability comes primarily from its hybrid design: CNN captures meaningful local cues from web artifacts, while BiLSTM handles context and sequence things like redirect paths and changing DOM structures that spoofing campaigns typically exploit. Figure 4 confirms that, indeed, PHISHCATCHER 2.0 is ready for real-world use, where attacks keep adapting and can move across sites.

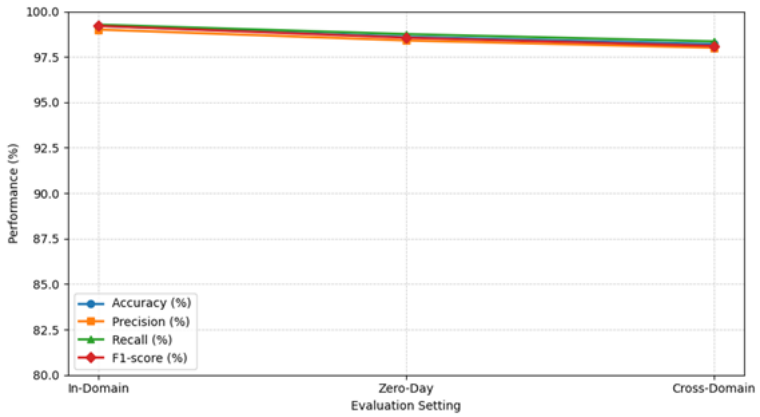

Fig. 4. Zero-day and cross-domain generalization analysis of PHISHCATCHER 2.0

F. Output Vizualization

In Figure 5 PHISHCATCHER 2.0 in action the real-time output interface for a real-time, adaptive, client-side defense against sophisticated web spoofing. You can directly test a URL: just paste the link, and instantly see a classification emanating from the integrated detection pipeline. The result will spell it out, in clear words, the security verdict-for example, Phishing-appended with a confidence visualization that makes the decision easy to comprehend. The interface will also be presenting dual outputs: traditional machine learning predictions and deep learning predictions, so users can also conduct comparisons of the model behavior while the system is running. Status messages pop up, like "ML model trained successfully" and "DL model trained successfully", to confirm that the detection engine is initialized and ready for continuous monitoring at the browser level.

Importantly, these outputs illustrate how PHISHCATCHER 2.0 can provide real-time actionable security feedback that is vital in the process of defeating phishing attacks and more advanced spoofing, such as brand impersonation and credential harvesting. The registration interface further highlights inherent user-management functionalities supporting controlled access and personalization in practical deployments. To put it concisely, the visualization confirms that PHISHCATCHER 2.0 is indeed not only a strong detector but also a complete client-side defense solution: fast, interpretable, and user-facing phishing protection integrated into real browsing experiences.

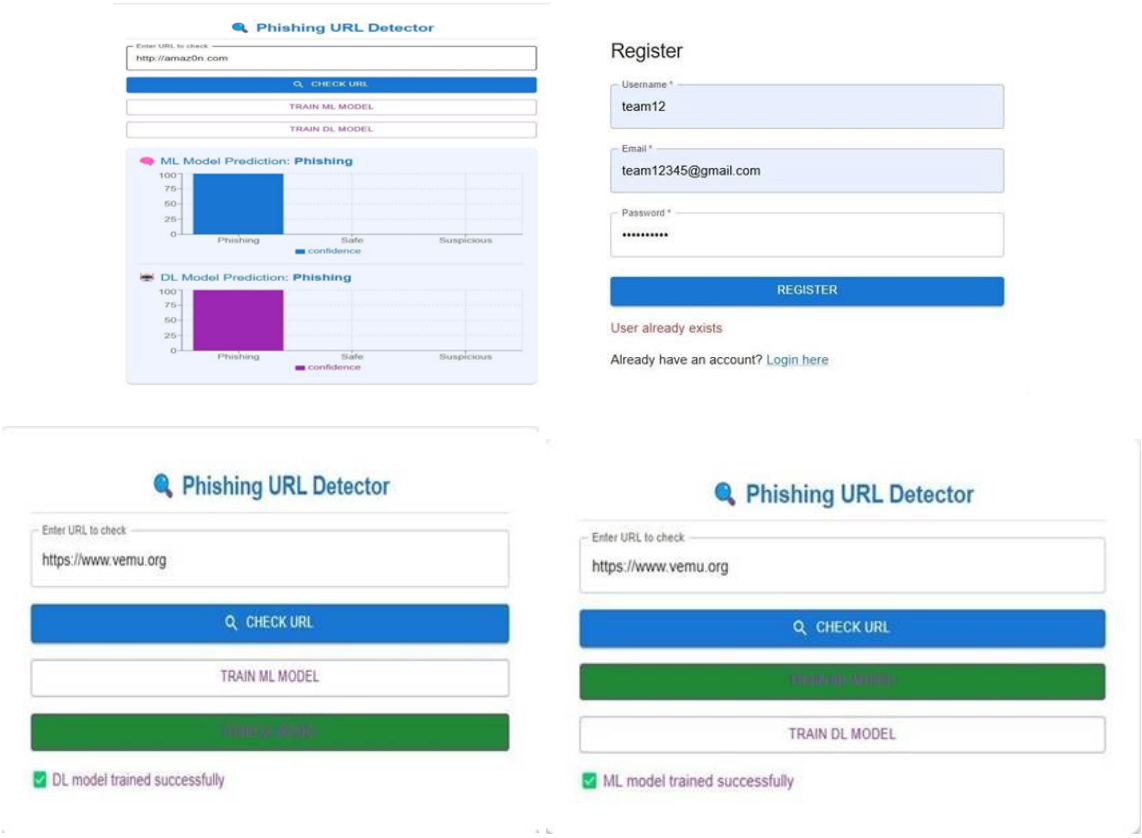

Fig. 5. Output interface of PHISHCATCHER 2.0 showing URL-based phishing detection

G. *Cloud–Assisted Learning*

Cloud Assisted Learning (CAL) will be integrated with PHISHCATCHER 2.0 as part of resistance training against constant phishing attacks. However, the classifier also uses a real-time CNN + BiLSTM learning solution on the browser level, and CAL introduces periodical adjustments by providing new spoofing examples to the global learning process as input. CAL thus aims at handling actual-world challenges such as zero-day phishing sites, UI mimicry attacks, Browser-in-the-Browser (BiTB) attacks, DOM injection, and redirection attacks, in which static-classifier-based systems have been known to deviate from their ideal performance paths oftentimes. In the CAL system, the spotted suspected events in the client are compressed into privacy-preserving feature vectors.

These contain URL lexical features, anomalous DOM tree patterns, script behavior features, and spoofing-specific contextual patterns. These compact models are transmitted securely to the cloud, and a centralized machine learning process there trains the model based on the aggregated threat data collected in the various networks spread across the globe. The weights are then transmitted back to the users as compact updates in PHISHCATCHER 2.0. Crucially, CAL enhances the detection pipeline by optimizing generalization performance on novel spoofing templates, without compromising the key benefit of client-execution: fast response time and privacy-friendly execution. In conclusion, CAL boosts PHISHCATCHER 2.0 from being a high-performing detector to an adaptive defense system that is able to maintain sustained performance against sophisticated and evolving web spoofing attacks.

H. *Ablation Study*

In Table 4, an ablation study has been done to examine the effect of each component in the PHISHCATCHER 2.0 – Real-Time Adaptive Client-Side Defense Against Sophisticated Web Spoofing Attacks system on performance. The combination of all components (CNN + BiLSTM + Cloud-Assisted Learning) achieves the highest value with an accuracy of 99.20%, precision of 99.00%, recall of 99.27%, and an F1-score of 99.20%, illustrating the effectiveness of the hybrid model in a real-world browsing experience. Removing the CAL component results in a reduction in performance with an accuracy and F1-score of 98.60%, indicating the significance of adaptive updates in continuing to deliver effectiveness in the face of evolving and zero-day attacks of web spoofing. Without the BiLSTM component, it will fallback on a CNN-only model, boasting an accuracy of 97.90% and F1 score of 97.85%, thus confirming that modeling sequences plays an important role in identifying spoofing attacks associated with multi-step redirects, transitions of sessions, and changes in DOM on-the-fly. However, performing BiLSTM only will lead to an accuracy of 97.30% and F1 score of 97.25%, thereby concluding that it overlooks important details encoded by CNN-driven local features associated with tiny URLs and anomalies at the DOM DOM-level.

When compared with the best traditional approach (accuracy of 97.80% and F1 score of 97.75% by XGBoost), there is definitely the benefit of combination in PHISHCATCHER 2.0, and this proves that all the modules work together effectively towards the goal of phishing webpage detection. In conclusion, from Table 3, the strength of PHISHCATCHER 2.0 lies in its combination of different modules, and not in the combination of any single module.

Table IV: Ablation Study of PHISHCATCHER 2.0 (CNN + BiLSTM)

Variant / Configuration	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Key Observation
Full Model: CNN + BiLSTM + CAL (PHISHCATCHER 2.0)	99.20	99.00	99.27	99.20	Best overall, strong spoofing robustness
CNN + BiLSTM (without CAL)	98.60	98.40	98.75	98.55	Lower zero-day/cross-domain adaptability
CNN Only (without BiLSTM)	97.90	97.60	98.10	97.85	Loses sequential spoofing/redirect learning
BiLSTM Only (without CNN)	97.30	97.10	97.55	97.25	Weak in local feature extraction (URL/DOM cues)
Traditional ML Baseline (Best: XGBoost)	97.80	97.60	97.90	97.75	Strong baseline but less spoof-aware

V. CONCLUSION AND FUTURE WORK

This research proposes PHISHCATCHER 2.0, an adaptive client-side defense solution that effectively counters clever web-spoofing attacks in real time and improves the efficiency of general phishing filters. PHISHCATCHER 2.0 employs a hybrid CNN-BiLSTM model to detect local spoofing features in URLs, Document Object Model patterns, and the content of injected scripts, as well as sequence spoofing features in redirect chains and page transitions. This creative design helps ensure the effective detection of sophisticated spoofing attacks and supports low-latency inferencing to enable seamless defense at the browser level. Experiments show that PHISHCATCHER 2.0 performs even better than the baseline models. The proposed Model achieves 99.20% accuracy, 99.00% precision, 99.27% recall, and 99.20% F1-score while maintaining the efficiency required for real-time defensive operations. Cloud-Assisted Learning is also found to improve the adaptive resistance property of PHISHCATCHER 2.0 against both zero-day attacks and concept drift, while satisfying the constraints of privacy-preserving adaptive learning and preventing increased computation cost on the client side. In the future, the research work to be carried out would focus on integrating multimodal visual analysis, improving interpretability, optimizing deployment on resource-limited devices, and conducting large-scale evaluation to improve the system's resilience in a constantly changing environment.

References

1. Bollu, J., Roseline, K. S., & Rakesh, C. (2025). Phish Catcher: Client-side defense against web spoofing attacks using machine learning. *Utilitas Mathematica, 126*, 1–15.
2. Baskota, S. (2025). *Phishing URL detection using bidirectional LSTM networks* (arXiv Preprint No. arXiv:2504.21049). arXiv.
3. Altan, A., Bachir, M., Parbhulkar, P., Rizvi, M., & Farazi, M. (2025). *Dual-path phishing detection integrating transformer-based NLP with structural URL analysis* (arXiv Preprint No. arXiv:2509.20972). arXiv.
4. Hossain, H., Al Arafat, A. A., Shepard, J., Craig, D., & Parvez, M. (2025). *A graph-attentive LSTM model for malicious URL detection* (arXiv Preprint No. arXiv:2510.15971). arXiv.

5. Vishal, P., Srinath, S., Adithya, S., & Kumar, R. (2025). Phishing detection using CNN and BiLSTM. *International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering, 13*(11), 45–52.
6. Rahman, M. A., Ahmed, T., & Islam, S. H. (2025). Lightweight malicious URL detection using deep learning-based hybrid CNN–BiLSTM architecture. *Scientific Reports, 15*, Article 26653.
7. Singh, R., & Verma, A. (2025). Real-time client-side phishing detection using ensemble machine learning models. *International Journal of Information Technology and Computer Engineering, 9*(5), 22–31.
8. Roy, S., Saha, S., & Nilizadeh, S. (2024). *PhishLang: A real-time, fully client-side phishing detection framework using MobileBERT* (arXiv Preprint No. arXiv:2408.05667). arXiv.
9. Thaqi, L., Halili, A., Vishi, K., & Rexha, B. (2024). *NoPhish: Efficient Chrome extension for phishing detection using machine learning techniques* (arXiv Preprint No. arXiv:2409.10547). arXiv.
10. Hoad, T., & Karafili, E. (2024). A web browser plugin for users' security awareness. In *Proceedings of the 19th International Conference on Availability, Reliability and Security* (pp. 1–7). ACM.
11. Asiri, S., Xiao, Y., & Alzahrani, S. (2024). Towards improving phishing detection system using human-in-the-loop deep learning model. In *Proceedings of the 2024 ACM Southeast Conference* (pp. 77–85). ACM.
12. Asiri, S., Xiao, Y., Alzahrani, S., & Li, T. (2024). PhishingRTDS: A real-time detection system for phishing attacks using a deep learning model. *Computers & Security, 141*, Article 103843.
13. Wang, M., Song, L., Li, L., Zhu, Y., & Li, J. (2024). Phishing webpage detection based on global and local visual similarity. *Expert Systems with Applications, 252*, Article 124120.
14. Varshney, G., Raj, A., Sangwan, D., Abuadbba, S., Mishra, R., & Gao, Y. (2025). A login page transparency and visual similarity-based zero-day phishing defense protocol. *Computers & Security*, Article 104598.
15. Ji, F., Lee, K., Koo, H., You, W., Choo, E., Kim, H., & Kim, D. (2025). Evaluating the effectiveness and robustness of visual similarity-based phishing detection models. In *Proceedings of the 34th USENIX Security Symposium (USENIX Security 25)* (pp. 3201–3220). USENIX Association.
16. Murhej, M., & Nallasivan, G. (2025). Component features-based enhanced phishing website detection system using EfficientNet, FH-BERT, and SELU-CRNN methods. *Frontiers in Computer Science, 7*, Article 1582206.
17. Sruthi, K., & Naik, S. M. (2025). A novel framework for phishing attack detection using domain-adapted GloVe embeddings and attention-enhanced neural sequence model. *Applied Soft Computing*, Article 114441.
18. Ahmed, S. T., Kumar, V. V., Singh, K. K., Singh, A., Muthukumaran, V., & Gupta, D. (2022). 6G enabled federated learning for secure IoMT resource recommendation and propagation analysis. *Computers and Electrical Engineering*, *102*, 108210.
19. Sathiyamoorthi, V., Ilavarasi, A. K., Murugeswari, K., Ahmed, S. T., Devi, B. A., & Kalipindi, M. (2021). A deep convolutional neural network based computer aided diagnosis system for the prediction of Alzheimer's disease in MRI images. *Measurement*, *171*, 108838.
20. Ahmed, S. T., Vinoth Kumar, V., Mahesh, T. R., Narasimha Prasad, L. V., Velmurugan, A. K., Muthukumaran, V., & Niveditha, V. R. (2024). FedOPT: federated learning-based heterogeneous resource recommendation and optimization for edge computing. *Soft Computing*, 1-12.