IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 9, September 2026

DOI: 10.17148/IJARCCE.2026.15902

Lightweight and Scalable Real-Time Phishing URL Detection Models: A Comparative Review and Proposed Framework

Abhishek Kumar[1], Anand Kumar[2]

Department of MCA, Babasaheb Bhimrao Ambedkar Bihar University, Muzaffarpur, Bihar India[1,2]

Abstract: Phishing URL detection has advanced considerably through machine learning and deep learning, with recent published systems reporting accuracy figures that frequently exceed 95%. However, a close reading of this literature shows that high accuracy is often achieved using large feature sets, deep sequence architectures, or ensembles comprising a dozen or more classifiers — approaches that are difficult to justify on resource-constrained platforms such as browser extensions, mobile applications, or edge gateways, where real-time response and a small memory footprint matter as much as raw detection accuracy. This paper reviews six recent studies on machine-learning-based phishing URL detection, comparing their algorithms, feature counts, and reported accuracy, and identifies a consistent gap: none of the reviewed works measure deployment cost — inference latency, model size, or throughput under concurrent load — alongside their accuracy claims. Building on this analysis, the paper proposes a tentative methodology for a lightweight, scalable, real-time phishing-detection framework that restricts features to those computable from the URL string alone, benchmarks compact classical classifiers against the heavier baselines identified in the review, and treats latency, model size, and throughput as first-class evaluation metrics rather than afterthoughts. The proposed direction is intended to narrow the gap between laboratory-grade accuracy and field-deployable, real-time phishing protection.

I. INTRODUCTION

Phishing continues to be one of the most common vectors for credential theft and financial fraud, and URL-based detection remains the first line of defense in browsers, email gateways, and network security tools [1], [5]. Over the past several years, the dominant research trend has been to chase ever-higher accuracy — through deeper neural architectures, larger feature vectors, or ensembles combining many classifiers — and the six studies reviewed in this paper are representative of that trend, reporting accuracies ranging from 88% to over 99.9% [1]-[6].

What is comparatively under-examined is the cost of achieving that accuracy in a deployment setting. A browser extension or a mobile app that must classify a URL before a page finishes loading has a latency budget measured in milliseconds; a network edge device may have only a few hundred megabytes of memory available for a detection model; a high-traffic email gateway must sustain thousands of classifications per second. None of these constraints are captured by accuracy, precision, recall, or F1-score alone. This paper therefore reviews the six studies with a specific lens — feature-extraction cost, model complexity, and reported (or unreported) real-time performance — and uses the resulting gap analysis to motivate a lightweight and scalable detection framework.

A. Motivation

Several of the reviewed systems explicitly use the words "real-time" in their title or abstract [1], [5], yet none report the actual time taken to classify a single URL end-to-end, including feature extraction. Several rely on features that require live WHOIS lookups, DNS resolution, or third-party page-rank/Google-index queries [5], [6] — network round-trips that can add hundreds of milliseconds per URL regardless of how fast the underlying classifier is. This mismatch between the stated goal (real-time detection) and the evaluation performed (offline accuracy on a static test split) is the central motivation for the present work.

II. RELATED WORK

Baskota **[1]** proposes a Bidirectional LSTM model for four-class URL classification (benign, phishing, defacement, malware) trained on a dataset of over 650,000 URLs, tokenized at the character level, reporting 97-98% overall accuracy along with a Flask-based web interface for single-URL scanning. Character-level Bi-LSTM sequence models carry a non-trivial parameter count and per-sample inference cost, but the paper does not report model size, memory usage, or per-prediction latency, leaving its suitability for constrained or high-throughput real-time environments unquantified.

DOI: 10.17148/IJARCCE.2026.15902

Parimi et al. **[2]** present URLDefend, a system that trains and compares thirteen machine learning classifiers — including Logistic Regression, Random Forest, Gradient Boosting, CatBoost, and a Multi-Layer Perceptron — alongside a dedicated typosquatting-detection module, with a React front end, a Flask back end, and Redis-based caching to reduce redundant DNS lookups. Their hyper-tuned Random Forest reaches the highest accuracy at 98%. While the explicit use of caching shows some deployment awareness, maintaining thirteen candidate models in production increases operational complexity, and the paper does not benchmark response time or throughput under concurrent user load.

Devi N. et al. **[3]** take a comparatively lightweight approach, applying Logistic Regression and Linear Discriminant Analysis over only thirteen features selected through Principal Component Analysis and Factor Analysis, combined with K-Means clustering for low/medium/high risk tiering. Logistic Regression achieves 88.3% accuracy — the lowest of the six studies reviewed — but its small feature set and simple linear decision boundary are inherently cheaper to compute than a deep or ensemble model, making this work a useful lower-complexity reference point for the accuracy-versus-efficiency trade-off explored in this paper.

Saxena, Degadwala, and Joshi **[4]** conduct a broader literature review spanning fifteen phishing-detection studies published between 2014 and 2024, and explicitly identify "high computational cost limits real-time deployment" as a recurring limitation among ensemble and deep-learning approaches in that body of work. Their proposed methodology is intentionally tentative, following a standard pipeline of data collection, preprocessing, feature selection, model training, and evaluation, but it stops short of specifying concrete lightweight design choices, latency targets, or a deployment architecture — a gap this paper aims to fill.

Rehman et al. **[5]** evaluate five classical classifiers — k-Nearest Neighbors, Naive Bayes, Decision Tree, Random Forest, and Random Tree — on the 235,795-instance, 54-attribute UCI PhiUSIIL dataset, with the Random Forest and Decision Tree models both exceeding 99.9% accuracy. Although the paper's title emphasizes real-time detection, the 54-attribute feature vector includes several attributes that in practice require WHOIS records, DNS resolution, or webpage content parsing to compute; this feature-extraction cost — often the true bottleneck in a real-time pipeline rather than model inference itself — is not measured or discussed.

Yi, Omotosho, and Balogun **[6]** evaluate Random Forest, XGBoost, and LightGBM across four independent benchmark datasets using an 87-feature set, reporting an average cross-dataset accuracy of approximately 96% and, notably, the strongest generalization performance among the six reviewed works. Their use of SHAP analysis further identifies google_index and page_rank as consistently influential features, and the trained models are deployed through a Flask web application. Despite this deployment focus, the paper does not report inference latency, memory footprint, or throughput, and — as with [5] — a meaningful share of the 87 features depend on live web or DNS queries rather than the URL string alone.

III. IDENTIFIED RESEARCH GAP

Synthesizing the six studies reveals a consistent pattern. First, "real-time" and "lightweight" are used descriptively rather than measured: not one of the six papers reports inference latency, serialized model size, memory footprint, or throughput under concurrent load — accuracy, precision, recall, and F1-score are treated as sufficient proxies for deployability, which they are not. Second, feature-set size varies widely across the six works (from 13 features in [3] to 87 in [6]) without any of the papers studying the accuracy-per-feature trade-off directly, making it difficult for a practitioner to choose a feature set appropriate to a given latency budget. Third, several of the highest-accuracy systems [5], [6] depend on features requiring live WHOIS, DNS, or third-party ranking/indexing lookups, which introduce network latency and external service dependency that a genuinely real-time, edge-deployable system should avoid or explicitly account for. These three observations together define the gap that the proposed framework, described in Section IV, is intended to address.

IV. PROPOSED METHODOLOGY (TENTATIVE)

A. Data Collection

The proposed framework will draw on the same class of publicly available labeled URL datasets used across the reviewed literature — including Kaggle phishing/benign URL collections referenced in [1] and the UCI PhiUSIIL dataset used in [5] — combined into a single benchmark set with a balanced phishing/legitimate split, consistent with the balanced-dataset practice already followed in [3] and [6].

DOI: 10.17148/IJARCCE.2026.15902

B. Lightweight Feature Selection

In direct response to the gap identified in Section III, feature engineering will be restricted to attributes computable from the URL string alone — length, character and digit counts and ratios, special-character frequency, subdomain count, and similar lexical measures — deliberately excluding features that require WHOIS, DNS, or third-party page-rank/indexing queries. This removes the network round-trip cost that affects [5] and [6] and keeps end-to-end classification time bounded by local computation only.

C. Candidate Lightweight Models

Rather than defaulting to the largest available ensemble, the framework will benchmark compact classifiers — Logistic Regression, a depth-limited Decision Tree, a shallow Random Forest (bounded tree count and depth), and a reduced-estimator LightGBM configuration — directly against the heavier baselines reported in [1], [2], and [5], to quantify how much accuracy is actually sacrificed for a substantial reduction in model size and inference cost.

D. Real-Time and Scalability Evaluation Metrics

Beyond the conventional accuracy, precision, recall, and F1-score metrics used throughout the reviewed literature, the proposed evaluation will additionally report: (i) average inference latency per URL in milliseconds, including feature extraction; (ii) serialized model size in kilobytes/megabytes; (iii) peak memory usage during inference; and (iv) throughput in classifications per second under simulated concurrent load. Treating these as first-class metrics, rather than omitting them as [1]-[6] all do, is the central methodological contribution of this proposal.

E. Deployment Architecture

A lightweight REST endpoint (Flask or FastAPI, following the deployment pattern already established in [1], [2], and [6]) will expose the trained model for single-URL and batch prediction, sized and load-tested for browser-extension or edge-gateway deployment rather than only a general-purpose web demo.

V. COMPARATIVE ANALYSIS

Table I summarizes the six reviewed studies against the dimensions relevant to lightweight, real-time deployment. "Deployment Complexity" is a qualitative assessment based on model count, feature-extraction dependencies, and architecture, since none of the six papers report a quantitative complexity or latency measure directly — which is itself the gap motivating this review.

Ref.	Study / Model(s)	Feature Count	Reported Accuracy	Latency / Model Size Reported?	Deployment Complexity
[1]	Baskota — Bi-LSTM (4-class)	Char-level sequence (no fixed count)	97-98%	No	High (deep sequence model)
[2]	Parimi et al. — 13-model ensemble + typosquatting module	Lexical + domain + HTML	98% (tuned RF)	No	High (13 models + Redis cache)
[3]	Devi N. et al. — Logistic Regression / LDA	13 (PCA/FA-selected)	88.3%	No	Low
[4]	Saxena et al. — Literature review, tentative pipeline	Varies by cited study	N/A (review)	No	N/A
[5]	Rehman et al. — RF / Decision Tree (UCI PhiUSIIL)	54	99.99%	No	Medium-High (WHOIS/DNS-dependent features)
[6]	Yi, Omotosho & Balogun — RF / XGBoost / LightGBM	87	~96% (cross-dataset)	No	Medium-High (web/DNS-dependent features)

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 9, September 2026
DOI: 10.17148/IJARCCE.2026.15902

VI. CONCLUSION AND FUTURE WORK

This review of six recent phishing URL detection studies shows a field that has optimized heavily for accuracy — several systems now exceed 97% — while largely overlooking the deployment-side metrics (latency, model size, memory, throughput) that determine whether a model is actually usable in a real-time, resource-constrained setting. The proposed tentative framework addresses this gap by restricting features to those computable without network lookups, benchmarking compact classifiers against the heavier baselines identified in the literature, and adopting latency, size, and throughput as explicit evaluation criteria. Future work will implement this pipeline, report the resulting accuracy-versus-efficiency trade-off curve, and validate the framework under simulated concurrent load to assess its suitability for browser-extension and edge deployment.

REFERENCES

[1]. S. Baskota, "Phishing URL Detection using Bi-LSTM," arXiv:2504.21049v1 [cs.CR], Apr. 2025.
[2]. N. Parimi, Y. Padamati, N. M. Dindigalla, S. K. Chodisetti, and S. P. Bulla, "Urldefend: Reinventing Phishing Detection With Machine Learning Models," Int. J. Creative Res. Thoughts (IJCRT), vol. 13, no. 3, pp. j757-j764, Mar. 2025.
[3]. Y. Devi N., A. C, and H. S. P., "Phishing URL Detection using Machine Learning," Int. J. Res. Trends Innov. (IJRTI), vol. 10, no. 9, pp. a434-a440, Sep. 2025.
[4]. D. Saxena, S. Degadwala, and M. Joshi, "Phishing URL Detection Using Machine Learning," Int. J. Sci. Res. Sci. Technol. (IJSRST), vol. 13, no. 1, pp. 19-25, Jan.-Feb. 2026.
[5]. A. U. Rehman, I. Imtiaz, S. Javaid, and M. Muslih, "Real-Time Phishing URL Detection Using Machine Learning," Eng. Proc., vol. 107, no. 108, Sep. 2025.
[6]. L. Yi, A. Omotosho, and H. Balogun, "Phishing URL Detection and Interpretability With Machine Learning: A Cross-Dataset Approach," Security and Privacy, vol. 9, e70175, 2026.

