Bulletin of Electrical Engineering and Informatics
Vol. 14, No. 6, December 2025, pp. 4876~4889
ISSN: 2302-9285, DOI: 10.11591/eei.v14i6.10099

Real-time browser-integrated phishing uniform resource locator detection via deep learning and fuzzy matching

Dam Minh Linh[1], Han Minh Chau[2], Huynh Trong Thua[1], Tran Cong Hung[3]
[1]Information Security Technology Lab, Faculty of Information Technology, Posts and Telecommunications Institute of Technology, Ho Chi Minh City, Vietnam
[2]Faculty of Information Technology, HUTECH University, Ho Chi Minh City, Vietnam
[3]Faculty of Computer Science and Engineering, The Saigon International University, Ho Chi Minh City, Vietnam

Article Info

Article history:

Received Feb 16, 2025
Revised Sep 29, 2025
Accepted Oct 14, 2025

Keywords:

Blacklist and whitelist URLs
Character vocabulary
Deep neural networks
Fuzzy string matching
Online URL classification
Phishing prevention

ABSTRACT

Phishing attacks through deceptive URLs remain a critical cybersecurity threat, particularly in financial transactions and online payment systems. This study evaluates multiple deep learning (DL) models on the PhiUSIIL dataset of 235,795 URLs, with bidirectional gated recurrent unit (BiGRU) achieving the best performance—99.82% accuracy at a 60:40 split, along with high precision, F1-score, and the lowest test loss. To further improve detection of obfuscated URLs, an enhanced BiGRU variant is proposed using an expanded 366-character vocabulary. For real-time deployment, a Chrome extension is developed, integrating exact and fuzzy matching via the Ratcliff–Obershelp algorithm with cloud-based whitelist and blacklist checks. When fuzzy matching is inconclusive, the BiGRU model performs the final classification. By combining an adaptive browser-side tool with a robust DL backend, the proposed system ensures high accuracy, scalability, and efficiency for phishing detection in practical web environments.

Corresponding Author:

Han Minh Chau
Faculty of Information Technology, HUTECH University
475A Dien Bien Phu Street, Ward 25, Binh Thanh District, Ho Chi Minh City, Vietnam
Email: hm.chau80@hutech.edu.vn

1. INTRODUCTION

Although URL-based phishing is not new to cybersecurity, it continues to grow in both complexity and frequency. Attackers increasingly employ obfuscation techniques and domain mimicry to bypass traditional blacklists and deceive even experienced users. Such attacks aim to trick victims into disclosing personal information, financial credentials, or authentication tokens through fraudulent links on online platforms, thereby enabling adversaries to gain unauthorized access and often exfiltrate funds or sensitive assets. These challenges necessitate the development of intelligent and adaptive models capable of accurately classifying URLs while maintaining minimal latency.

The Anti-Phishing Working Group (APWG) reported 877,536 phishing incidents in Q2 2024, marking a decrease from 963,994 in Q1 [1]. Despite the decline, attackers are diversifying their methods by leveraging phone calls and text messages in addition to email, mainly targeting banking and payment service users. Social media platforms remain the most impacted sector, accounting for 32.9% of all attacks. Illustrating the real-world consequences of such attacks, a recent report published in February 2024, Pepco Group, a prominent European retail conglomerate owning brands such as Pepco, Dealz, and Poundland, fell victim to a sophisticated phishing attack targeting its Hungarian branch, resulting in a financial loss of approximately 15.5 million euros (equivalent to 16.8 million USD) [2].

The phishing URL detection method based on a whitelist was proposed by Azeez *et al.* [3], achieving 96.17% accuracy. These were further improved in [4] through dynamic list management, which integrated blacklists, whitelists, and greylists. While list-based methods are efficient in processing time when comparing URLs against known entries, they remain limited in detecting newly crafted or obfuscated phishing URLs. To address this, machine learning (ML) techniques have been explored. Sabahno and Safara [5] applied the improved spotted hyena optimization algorithm for feature selection in a support vector machine (SVM) model, reaching 98.64% accuracy on 11,055 URLs. Similarly, Kara *et al.* [6] used 11 handcrafted features with a random forest (RF) model, achieving 98.90% accuracy on 32,928 samples. However, these ML approaches may still lose important structural or semantic information when handling long URLs, which can affect classification performance.

Awasthi and Goel [7] evaluated ensemble methods on three datasets comprising 13,511 URLs, with Bagging, AdaBoost (AB), XGBoost, and voting ensemble achieving high accuracy; the best model reached 99.18%. Meanwhile, Aljabri *et al.* [8] compared RF, Naïve Bayes (NB), convolutional neural network (CNN), and long short-term memory (LSTM) on 66,506 URLs, reporting that NB performed best with 96.01% accuracy. In another study, Alqahtani *et al.* [9] proposed a phishing detection method using a deep autoencoder classifier, achieving up to 99.28% accuracy. Similarly, Bahaghighat *et al.* [10] developed an ML-based predictive framework with six algorithms, where XGBoost achieved the highest accuracy of 99.2% on a dataset of more than 88,000 URLs.

Pavani *et al.* [11] conducted a comprehensive evaluation of ten classical ML algorithms, including RF, decision tree (DT), logistic regression (LR), SVM, Gaussian Naïve Bayes (GNB), k-nearest neighbors (KNN), Bernoulli Naïve Bayes (BNB), AB, neural networks (NN), and gradient boosting (GB). Among these, the RF achieved the highest accuracy of 98.55%. The integration of model interpretation techniques such as SHAP and LIME further enhanced transparency and interpretability. In a related study, Kalabarige *et al.* [12] employed a boosting-based stacked ensemble framework with hybrid feature selection strategies, reporting superior performance over base learners with classification accuracy ranging from 96.18% to 98.95%. The multilayered architecture significantly improved phishing detection capabilities, albeit at the expense of increased training time. Meanwhile, Asif *et al.* [13] proposed a hybrid model combining LSTM and CNN architectures with DeepWalk-based URL embeddings. Their approach achieved marginal gains in classification accuracy, but the improvements were not statistically significant.

Alqahtani and Abu-Khadrah [14] proposed a phishing website detection model that integrates three classification algorithms: RF, SVM, and Bagging, within a unified system. The model was evaluated on a dataset comprising 1,353 URLs, including 548 legitimate, 702 phishing, and 103 suspicious URLs. Experimental results demonstrated that the proposed ensemble approach achieved an overall accuracy of 92.33%, with a precision of 92.13%, recall of 92.09%, and F1-score of 92.10%.

Buu and Cho [15] proposed a fuzzy-calibrated transformer network for phishing URL detection. The model was evaluated on the ISCX-URL2016 dataset comprising 35,000 benign, 9,000 phishing, 11,000 malware, and 12,000 spam URLs, achieving 98.95% accuracy and 95.10% recall. However, the method introduces higher computational overhead due to fuzzy calibration and is constrained by the relatively small dataset size.

Phishing URL attacks are becoming increasingly sophisticated, posing serious risks to users, especially in online financial transactions. Traditional detection methods—such as whitelist/blacklist filtering and classical ML classifiers—often struggle to extract meaningful features from raw URL structures. Moreover, many DL models treat URLs as simple linear sequences, thereby overlooking critical structural information embedded in different URL components. The key contributions of this work are summarized as follows and validated through extensive empirical evaluation:

- To strengthen real-time security and safeguard user information—particularly in online financial transactions—a browser extension was developed in JavaScript and integrated into Chrome. Rather than embedding a DL model directly, the approach was advanced into a verification pipeline that combines cloud-based blacklist/whitelist checks via Firestore with an inspection mechanism using a pre-trained bidirectional gated recurrent unit (BiGRU) model to assess previously unseen URLs.
- The browser extension was further enhanced with the Ratcliff–Obershelp algorithm, which computes string similarity by recursively identifying the longest common subsequences between two input strings. This algorithm verifies URLs against two datasets: a whitelist of 3,000 benign URLs [16] and a blacklist of 2,500 phishing URLs [17]. Both datasets are hosted on the Cloud Firestore platform and shared with the cybersecurity community for collaborative use [18]. Communication with the endpoint is handled automatically through a REST API, enabling efficient real-time URL evaluation. This approach addresses several limitations noted in [3], [4] by moving beyond traditional blacklist–whitelist methods.
- Extends the character-level vocabulary from 270 [19] to 366 symbols and evaluates five deep learning (DL) models artificial neural network (ANN), CNN, LSTM, gated recurrent unit (GRU), and BiGRU—on

the academic dataset PhiUSIIL [20], which contains 235,795 URLs (134,850 benign and 100,945 phishing). Multiple train–test splits (5:5, 6:4, 7:3, 8:2, and 9:1) were applied to determine the optimal configuration for maximizing accuracy and minimizing misclassification.

– Addresses the limitations in [10], [14], [15]—including small dataset sizes, limited sequential modeling capacity, and insufficient evaluation diversity—by highlighting the superior performance of BiGRU. Capturing bidirectional sequence context, BiGRU outperformed the other models, achieving 99.82% testing accuracy at the 6:4 split and consistently exceeding 99.8% across all configurations when assessed with accuracy, precision, F1-score, loss, and confusion matrix.

The structure of the paper is as follows: section 2 presents the proposed algorithm, which combines Firestore URL management with a hybrid inspection procedure. Section 3 describes the evaluation methods and dataset, while section 4 reports and discusses the experimental results. Finally, section 5 concludes the study, followed by the acknowledgments and references.

2. PROPOSED ALGORITHM

2.1. Proposed batch URL bulk Firestore algorithm for Firestore URL management

To enable scalable and efficient management of phishing-related URL datasets, the batch URL bulk Firestore algorithm (BUBFA) is proposed. The algorithm automates batched ingestion of labeled URLs into Firestore collections, ensuring centralized storage of blacklist and whitelist entries. Firestore, a scalable NoSQL cloud database by Google, is widely used for real-time synchronization across web and mobile platforms. BUBFA accepts a newline-separated string of URLs, performs parsing and sanitization, and uploads them in bounded batches via HTTP POST requests. To comply with Firestore API rate limits, short delays are inserted between transmissions. Designed for real-time phishing detection frameworks, this mechanism ensures reliable, modular, and efficient integration of threat intelligence into a centralized repository.

BUBFA provides a structured mechanism for ingesting large sets of labeled URLs into Firestore in a reliable and rate-limited manner. It groups incoming URLs into bounded batches, performs sequential uploads, and ensures stable synchronization with the cloud database. Algorithm 1 presents the complete pseudocode of the proposed batching procedure.

Algorithm 1. Batch URL bulk Firestore algorithm
Input: A raw text string rawText containing newline-separated URLs
Output: URLs inserted into a Firestore collection, where collection ∈ {blacklist_urls, whitelist_urls}
1: urls ← Split rawText by newline characters, trim each entry, and remove empty lines
2: collection ← “blacklist_urls” or “whitelist_urls” ⟶ selected based on the dataset type
3: project_id ← “ptit-url-guardian”
4: firestore_url←“https://firestore.googleapis.com/v1/projects/”+project_id+“/databases/(default)/documents/”+collection
5: **for** i ← 0 to length(urls) step 50 **do**
6: batch ← urls[i : i + 50]
7: Call Send_Batch(batch)
8: Print “Uploaded i + batch.length of urls.length”
9: Sleep for 400 milliseconds ⟶ to avoid Firestore API rate limiting
10: **end for**
11: Print “Completed upload to Firestore collection:”, collection

Procedure: Send_Batch(batch)
1: **for** each url in batch **do**
2: body ← {
3: fields: {
4: url: { stringValue: url }
5: }
6: }
7: Send an HTTP POST request to firestore_url with body as payload
8: **end for**

a. Initialization phase (lines 1–4): the input string rawText is parsed into a list of trimmed, non-empty URLs. Depending on the labeling context, either the blacklist_urls or whitelist_urls Firestore collection is selected. In deployment, the whitelist contains approximately 3,000 URLs, while the blacklist holds about 2,500.

b. Batch processing and upload loop (lines 5–11): the cleaned URL list is partitioned into batches of 50. For each batch, the Send_Batch() procedure uploads the URLs to Firestore. To comply with Firestore API rate limits and avoid throttling, a fixed delay of 400 ms is introduced between consecutive uploads.
c. Send_Batch procedure (lines 1–8): the procedure iterates through each URL in the batch, creating a JSON document with the URL as a string field. Each document is uploaded to Firestore via an HTTP POST request. This stateless design ensures scalability and seamless integration of labeled threat data.

2.2. Hybrid URL inspection procedure

By leveraging a lightweight Chrome extension, the detection process starts by retrieving whitelist and blacklist data from Firestore. URLs are first checked using exact matching; if no direct match is found, the Ratcliff–Obershelp algorithm computes a similarity score through recursive pattern recognition of the longest common substrings, effectively identifying obfuscated or intentionally misspelled phishing URLs. Based on the similarity score and a predefined threshold, the URL is classified as either phishing or benign. When classification remains inconclusive, a BiGRU-based DL model provides the final decision. Real-time browser notifications then alert users, ensuring robust phishing protection directly at the point of access.

MSPGA coordinates multi-stage phishing detection directly within the browser by integrating whitelist/blacklist retrieval, fuzzy similarity scoring, and deep-learning-based URL classification. The mechanism continuously monitors browser navigation events, evaluates each URL through exact and fuzzy matching, and invokes a BiGRU model when a decision cannot be reached deterministically. Algorithm 2 presents the full pseudocode describing the operational flow of the MSPGA procedure.

Algorithm 2. MSPGA-multi-stage PhishGuard algorithm
Input: URL events captured from browser navigation and tab updates
Output: URL label ∈ {phishing, benign}
1: Initialize Firebase App with ptit-url-guardian credentials
2: Initialize db ← Firestore instance
3: Set threshold ← 0.85 for similarity-based phishing classification
4: Register listeners for onBeforeNavigate and onUpdated browser events
5: **for** each detected event **do**
6: url ← Extract URL from browser event
7: Call analyzeURL(url)
8: **end for**

Procedure: analyzeURL(url)
1: Print "Checking URL:", url
2: {blacklist, whitelist} ← fetchURLsFromFirebase()
3: **if** url ∈ blacklist **then**
4: notifyUser("phishing", url)
5: return
6: **end if**
7: **if** url ∈ whitelist **then**
8: notifyUser("benign", url)
9: return
10: **end if**
11: Print "No exact match → computing fuzzy similarity"
12: score_phish ← safeMaxScore(url, blacklist, "phishing")
13: score_benign ← safeMaxScore(url, whitelist, "whitelist")
14: Print scores for debugging
15: **if** score_phish ≥ threshold and score_phish > score_benign **then**
16: notifyUser("phishing", url, isSimilar=true)
17: **else if** score_benign ≥ threshold and score_benign > score_phish **then**
18: notifyUser("benign", url, isSimilar=true)
19: **else**
20: prediction = predictWithDeepModel(url)
21: notifyUser(prediction.label, url, modelUsed = true)
22: **end if**

Procedure: fetchURLsFromFirebase()
1: **Start timer**
2: phishing_snap ← db.collection("blacklist_urls").get()
3: whitelist_snap ← db.collection("whitelist_urls").get()
4: phishing ← Extract URL field from phishing_snap

5: whitelist ← Extract URL field from whitelist_snap
6**: End timer**
7: return {phishing, whitelist}
Procedure: safeMaxScore(url, list, label)
1: maxScore ← 0.0
2: **for** each item in list **do**
3: score ← ratcliffObershelp(url, item)
4: Print score between url and item
5: **if** score > maxScore **then**
6: maxScore ← score
7: **end if**
8: **end for**
9: return maxScore
Procedure: ratcliffObershelp(s1, s2)
1: **if** s1 or s2 is empty **then** return 0.0
2: lcs ← longestCommonSubstring(s1, s2)
3: **if** lcs is empty **then** return 0.0
4: Calculate left and right segments excluding lcs
5: Recursively compute similarity on segments
6: Return normalized score based on LCS and recursion
Procedure: predictWithDeepModel(url)
1: vectorized_url ← preprocessURL(url)
2: prediction ← deepModel.predict(vectorized_url)
3: if prediction ≥ 0.5 then
4: return {label: "phishing"}
5: else
6: return {label: "benign"}
Procedure: notifyUser(status, url, isSimilar=false)
1: **if** status = "phishing" **then**
2: Set badge ← "**!**", red
3: Show Chrome notification with warning

a. Initialization phase (lines 1–4): the detection module initializes a Firebase App instance with the ptit-url-guardian project credentials and establishes a Firestore database connection. A similarity threshold of 0.85 is defined for fuzzy-matching classification. Event listeners are then registered for two browser APIs—onBeforeNavigate and onUpdated—enabling the system to capture URL events during both page navigations and tab updates.
b. URL event handling loop (lines 5–8): for each detected browser URL event, the system extracts the current URL and invokes the main analysis procedure analyzeURL(url) to perform classification.
c. URL analysis procedure (lines 1–22):

- The procedure begins by logging the inspected URL and retrieving the latest blacklist and whitelist collections from Firestore via fetchURLsFromFirebase(). It then performs exact matching: if the URL is found in the blacklist, it is immediately flagged as phishing and the process terminates; if it appears in the whitelist, it is labeled as benign and terminates. When no exact match is found, the procedure computes fuzzy similarity scores using the safeMaxScore function, which applies the Ratcliff–Obershelp algorithm to compare the input URL against both lists. Finally, the computed similarity scores for phishing and benign categories are printed for debugging.
- If the phishing similarity score exceeds the threshold and is greater than the benign score, the URL is flagged as phishing with isSimilar set to true. Conversely, if the benign score meets the threshold and surpasses the phishing score, the URL is classified as benign. When neither score provides a definitive decision, the algorithm invokes the DL model predictWithDeepModel(url) based on the BiGRU architecture to predict the label. The final result is then returned to the user, together with an indication that the model was applied.
- Procedure: `fetchURLsFromFirebase()` (lines 1–7): this procedure starts a timer and retrieves snapshots from the blacklist_urls and whitelist_urls collections in Firestore. The extracted URLs are stored in two lists representing phishing and benign entries. Once completed, the timer stops and both lists are returned, ensuring the system operates with the most recent remote data.

d. Procedure: `safeMaxScore(url, list, label)` (lines 1–9): this function computes the maximum similarity score between the input URL and all entries in a given reference list using the `Ratcliff-Obershelp` algorithm. It initializes `maxScore` to zero, iterates through the list, and prints each computed score for debugging. After all comparisons, it returns the maximum score, reflecting how closely the URL matches known entries.
e. Procedure: `ratcliffObershelp(s1, s2)` (lines 1–6): this recursive function implements the Ratcliff–Obershelp algorithm for string similarity. It returns zero if either input string is empty or if no longest common substring (LCS) is found. Otherwise, it recursively calculates similarity scores for segments adjacent to the LCS. The final score is normalized to reflect structural similarity, making the method effective for detecting obfuscated or misspelled phishing URLs.
f. Procedure: `predictWithDeepModel(url)` (lines 1–6): this procedure preprocesses and vectorizes the input URL for a trained BiGRU-based DL model. The model outputs a phishing likelihood score, which is compared against a 0.5 threshold to assign a phishing or benign label. It classifies previously unseen URLs, thereby improving detection robustness.
g. Procedure: `notifyUser(status, url, isSimilar=false)` (lines 1–3): based on the classification result, this procedure updates the browser interface to inform the user. For URLs labeled as phishing, a red badge with an exclamation mark ("!") is set, and a Chrome notification with a warning message is displayed in real time to enhance user awareness and security.

3. EVALUATION METHODS AND DATASET

3.1. Hyperparameter selection for deep neural networks

To control for architectural bias and isolate the impact of different sequence modeling mechanisms, five DL models—ANN, CNN, LSTM, GRU, and the proposed BiGRU —were systematically designed under harmonized hyperparameter constraints. All models received the same character-level input with a fixed sequence length of 54, derived from a unified vocabulary of 366 tokens. Each model incorporated a 128-dimensional embedding layer to ensure consistent input representation. The primary architectural differences lie in their sequential and convolutional processing components. Table 1 summarizes the internal structure of each model.

Table 1. Summary of the neural network architectures and hyperparameters

Model	Input layer	Core layers	Dropout rate (s)	Dense layer (s)	Output layer
ANN	Embedding (128-dim)	Flatten→Dense(256)→Dense(128)→Dense(64)	0.4→0.3→0.2	3 layers (ReLU)	Dense(1, Sigmoid)
CNN	Embedding (128-dim)	Conv1D(256, k=3)→Conv1D(128, k=3)→GlobalMaxPooling1D	0.3→0.3→0.4	Dense(128, ReLU)	Dense(1, Sigmoid)
LSTM	Embedding (128-dim)	LSTM(128)	0.3→0.2	Dense(64, ReLU)	Dense(1, Sigmoid)
GRU	Embedding (128-dim)	GRU(128, return_sequences=True)→GRU(128)	0.3→0.4→0.4	Dense(128, ReLU)	Dense(1, Sigmoid)
BiGRU	Embedding (128-dim)	Bidirectional GRU(128, return_sequences=true)→bidirectional GRU(128, return_sequences=true)→attention layer	0.3→0.3	Dense(64, ReLU)→Dense(32, ReLU)	Dense(1, Sigmoid)

All models were trained using the Adam optimizer and binary cross-entropy loss, with early stopping applied at a patience of 5. A fixed batch size of 32 was used, and the output layer of each model employed a sigmoid activation function for binary classification between phishing and benign URLs. These standardized hyperparameters ensured a controlled and unbiased experimental setup, enabling rigorous and fair comparisons across all neural network architectures.

3.2. Extended character vocabulary and indexing method

The 270-character encoder is expanded by incorporating 96 additional symbols, yielding a total of 366 unique character encodings. These are organized into three functional categories to capture common obfuscation and semantic patterns in phishing URLs, as illustrated in Figure 1. This targeted expansion enhances the model's ability to generalize across multilingual, encoded, and adversarial URL structures.

a. URL-specific special characters (codes 271–291; 21 characters): include commonly used symbols in obfuscated or malicious URLs, such as \, ^, ~, ©, ×, ÷, and other extended Unicode punctuation. These characters enable the model to capture structural manipulations within path or parameter segments.

b. Multilingual and semantic characters (codes 292–345; 54 characters): consist of characters from Cyrillic, Chinese, Japanese (Kana), and Korean scripts, as well as extended Latin symbols (e.g., i, °F, Ḃ, and €) and invisible Unicode modifiers (\u00A0, \u200B, and \u2060). This allows the model to detect IDN-based spoofing and cross-script phishing attacks.
c. Cybersecurity-relevant keywords and domains (codes 346–366; 21 characters): contain domain suffixes (e.g., .com, .php, and .html) and query tokens (e.g., auth=, form=, and submit=) that frequently appear in phishing payloads or redirection URLs.

{'\\': 271, '^': 272, '~': 273, '`': 274, '©': 275, '®': 276, '™': 277, '§': 278, 'µ': 279, '¶': 280, '±': 281, '°': 282, '×': 283, '÷': 284, 'А': 285, 'Б': 286, 'В': 287, 'Г': 288, 'Д': 289, 'Е': 290, 'Ж': 291, 'З': 292, 'И': 293, 'Й': 294, 'К': 295, 'Л': 296, 'М': 297, 'Н': 298, 'О': 299, 'П': 300, 'Р': 301, 'С': 302, 'Т': 303, 'У': 304, 'Ф': 305, 'Х': 306, 'Ц': 307, 'Ч': 308, '汉': 309, '字': 310, 'ひ': 311, 'ら': 312, 'が': 313, 'な': 314, 'カ': 315, 'タ': 316, '한': 317, '글': 318, 'a': 319, 'e': 320, 'o': 321, 'p': 322, 'c': 323, 'y': 324, 'i': 325, 'ii': 326, 'iii': 327, '°F': 328, '°C': 329, 'Ḃ': 330, '£': 331, '₩': 332, '¥': 333, '€': 334, '\u00A0': 335, '\u200B': 336, '\u2060': 337, '.com': 338, '.net': 339, '.org': 340, '.gov': 341, '.edu': 342, '.io': 343, '.php': 344, '.asp': 345, '.html': 346, '.aspx': 347, 'api=': 348, 'auth=': 349, 'session=': 350, 'key=': 351, 'action=': 352, 'submit=': 353, 'form=': 354, 'secure=': 355, 'url=': 356, 'ref=': 357, 'click=': 358, 'next=': 359, 'from=': 360, 'to=': 361, 'path=': 362, 'dest=': 363, 'redir=': 364, 'access=': 365, 'reset=': 366}

Figure 1. Character vocabulary indexing scheme

3.3. Evaluation methodology

Model effectiveness is quantitatively assessed using a comprehensive set of evaluation metrics, including the confusion matrix (CMa), accuracy (Acc), precision (Pre), F1-score ($F1$), and loss (Los), as referenced in [21]–[24]. The confusion matrix, defined in (1), captures the relationship between predicted and actual class labels. Together, these metrics provide a holistic view of classification performance, enabling rigorous evaluation of both ML and DL models.

$$CMa = \begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix} \tag{1}$$

Here, TP denotes a true positive, i.e., phishing URLs correctly classified as phishing; TN denotes a true negative, i.e., benign URLs correctly classified as benign; FP refers to a false positive, i.e., benign URLs incorrectly classified as phishing; and FN refers to a false negative, i.e., phishing URLs incorrectly classified as benign. The remaining evaluation metrics are defined as (2)-(5):

$$Acc = \frac{TP+TN}{TP+TN+FP+FN} \tag{2}$$

$$Pre = \frac{TP}{TP+FP} \tag{3}$$

$$Rec = \frac{TP}{TP+FN} \tag{4}$$

$$F1 = 2 \times \frac{Pre \times Rec}{Pre \times Rec} \tag{5}$$

Here, Acc represents the proportion of correct predictions relative to the total number of samples, while Pre is defined as the ratio of TP to the sum of $TP + FP$. When Pre equals 1, the numerator and denominator are identical ($TP = TP + FP$), implying that FP=0. An increase in FP enlarges the denominator, thereby reducing precision. $F1$ is the harmonic mean of Pre and Rec, where a higher $F1$ value indicates that both Pre and Rec are high, reflecting improved classification performance, particularly when Rec approaches 1. Cross-entropy loss (Los) is a widely adopted objective function for binary classification tasks, including phishing versus legitimate URL detection [25]–[27]. Its mathematical formulation is given as (6):

$$Los = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)] \tag{6}$$

Here, N denotes the total number of samples, y_i is the ground-truth label of the i^{th} sample (1 for legitimate, 0 for phishing), and $\hat{y}_i$ is the predicted probability, typically generated by a sigmoid or softmax function. The logarithmic function uses the natural base e. Los quantifies the discrepancy between the actual and predicted labels, serving as the optimization objective during training. A lower Los indicates better model performance, whereas a higher Los suggests the need for further parameter tuning to enhance predictive accuracy.

3.4. Benchmark dataset and evaluation setup

This study employs the PhiUSIIL dataset [20], a benchmark specifically designed for classifying phishing and benign URLs. Published in a reputable cybersecurity journal (ISI Q1, impact factor=5.4, CiteScore=13.3), the dataset contains 235,795 URLs, including 134,850 benign and 100,945 phishing samples. The URLs were collected from diverse and credible sources to ensure both label accuracy and dataset representativeness.

To assess the impact of training data volume on model performance, the PhiUSIIL dataset is partitioned into five training–testing ratios ranging from 0.5 to 0.9. The test proportion in each case is calculated as $Testing\ (i) = 1 - Training\ (i)$, where i denotes the iteration. As shown in Table 2, increasing the training fraction reduces the test set size, enabling evaluation of model performance under varying levels of training data availability.

Table 2. Training–testing splits for the PhiUSIIL dataset containing 235,795 URLs

Simulation iteration (i)	Training fraction	Training URLs	Testing fraction	Testing URLs	Training–testing ratio
1	0.5	117,897	0.5	117,898	5:5
2	0.6	141,476	0.4	94,319	6:4
3	0.7	164,924	0.3	70,871	7:3
4	0.8	188,402	0.2	47,393	8:2
5	0.9	211,880	0.1	23,915	9:1

4. RESULTS AND DISCUSSION

All empirical evaluations were conducted on a high-performance server equipped with dual Intel Xeon E5-2696 v3 processors (2.30 GHz, 36 cores, and 72 threads), 64 GB DDR4 RAM, and an NVIDIA RTX 3090 XC3 Ultra Hybrid GPU (24 GB GDDR6X, 10,496 CUDA cores). To minimize implementation variance and ensure consistency, all models were executed in a standardized computing environment. Model performance was assessed using four quantitative metrics—accuracy, precision, F1-score, and loss—providing a comprehensive evaluation of predictive effectiveness and classification quality across ANN, CNN, LSTM, GRU, and the proposed BiGRU.

4.1. Real-time response detection in browser extensions

The browser extension, implemented in JavaScript, is enhanced into a lightweight Chrome tool for real-time phishing detection. It retrieves whitelist and blacklist data from Firestore, performs exact URL matching, and applies the Ratcliff–Obershelp algorithm for similarity scoring to identify obfuscated or deceptively modified phishing URLs. Figure 2 illustrates a browser-based security warning triggered when accessing a suspected phishing website. The notification, titled "website blocked warning", labels the site as "Phishing" in red, indicating potential malicious intent. The examined URL, https://celcom-life.weebly.com/, is flagged for possible impersonation aimed at stealing personal information. A red triangular icon with an exclamation mark reinforces the alert, accompanied by a cautionary message advising the user to return.

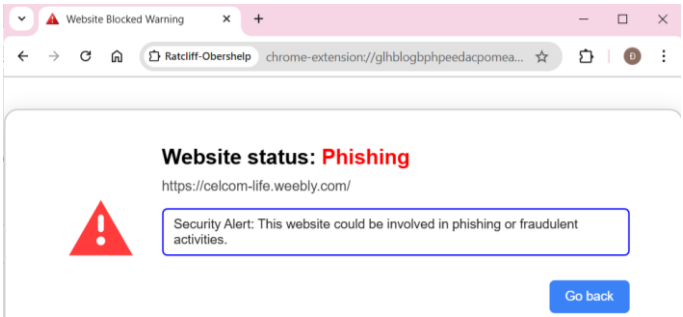

Figure 2. Browser extension supporting unsafe URL detection

Figure 3 illustrates the process of verifying a suspicious URL using the developed browser extension. The system queries the blacklist and whitelist from Firestore, and when no exact match is found, the fuzzy matching algorithm is triggered, producing similarity scores of 0.3836 with the phishing list and 0.2424 with the benign list. The total processing time is 1975.716 ms, demonstrating the system's capability to detect phishing URLs in near real-time directly within the browser.

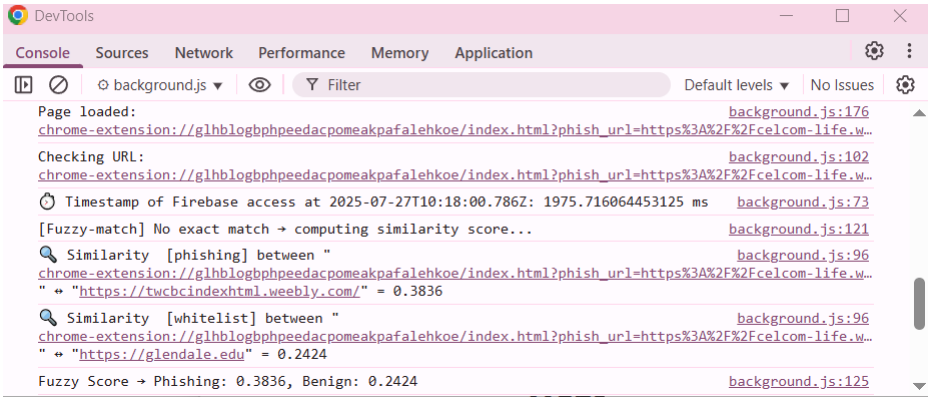

Figure 3. Console log illustrating the phishing detection process

4.2. Effectiveness of multi-model assessment

Table 3 shows that BiGRU consistently delivers high testing accuracy across all training–testing ratios, while ANN performs slightly lower. At the 6:4 ratio, BiGRU achieves the highest testing accuracy of 99.82%, compared to 99.81% for GRU, 99.80% for LSTM, and 99.79% for both ANN and CNN. From 5:5 to 9:1, BiGRU maintains a stable and superior accuracy trend, whereas CNN and ANN exhibit slight declines as the training portion increases. As illustrated in Figure 4, BiGRU demonstrates the best test accuracy distribution, reflected by its taller and more concentrated violin plot, while CNN records the lowest accuracy with a shorter and more dispersed distribution.

Table 3. Post-training and testing accuracies of the five algorithms evaluated on the PhiUSIIL dataset

Training–testing ratio	ANN		CNN		LSTM		GRU		BiGRU	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
5:5	0.99841	0.99783	0.99821	0.99804	0.99801	0.99787	0.99832	0.99785	0.99841	0.99802
6:4	0.99831	0.99795	0.99797	0.99787	0.99855	0.99808	0.99836	0.99810	0.99843	0.99819
7:3	0.99833	0.99790	0.99810	0.99802	0.99883	0.99819	0.99835	0.99820	0.99842	0.99809
8:2	0.99823	0.99773	0.99804	0.99775	0.99856	0.99798	0.99839	0.99796	0.99851	0.99802
9:1	0.99828	0.99770	0.99810	0.99770	0.99865	0.99783	0.99818	0.99792	0.99862	0.99796

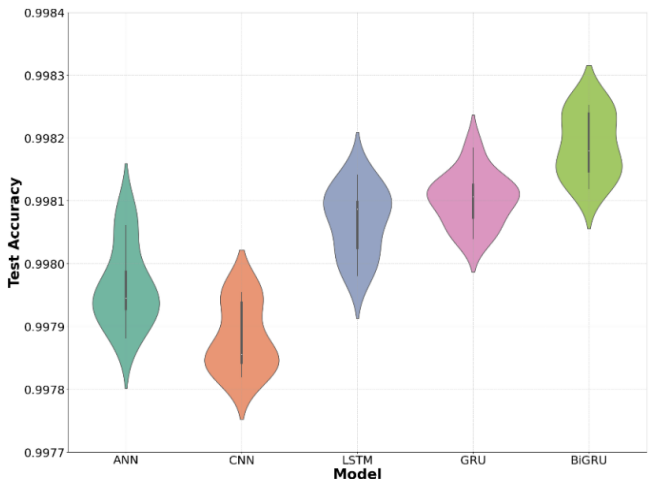

Figure 4. Violin plot of test accuracy distribution across models at a training–testing ratio of 6:4

Table 4 shows that the precision of all models remains generally stable across different training–testing ratios, with BiGRU and LSTM consistently yielding superior results. At the 6:4 ratio, BiGRU achieves the highest test precision at 99.7%, followed by LSTM and GRU at 99.694% and 99.691%, respectively, while ANN and CNN record the lowest values at 99.645% and 99.634%. As presented in Table 5, at the 7:3 training–testing ratio, GRU and LSTM achieve test F1-scores of 99.843% and 99.842%, respectively. BiGRU maintains consistently high performance across all ratios, peaking at 99.842% at the 6:4 split. In contrast, CNN and ANN exhibit slightly lower but more stable results, with F1-scores consistently around 99.8% across configurations.

Table 4. Precision of the five algorithms evaluated on the PhiUSIIL dataset

Training–testing	ANN		CNN		LSTM		GRU		BiGRU	
ratio	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
5:5	0.99723	0.99645	0.99688	0.99665	0.99654	0.99630	0.99754	0.99717	0.99772	0.99718
6:4	0.99705	0.99645	0.99646	0.99634	0.99749	0.99694	0.99718	0.99691	0.99728	0.99700
7:3	0.99711	0.99645	0.99715	0.99691	0.99801	0.99704	0.99713	0.99699	0.99733	0.99691
8:2	0.99694	0.99615	0.99672	0.99630	0.99750	0.99682	0.99722	0.99648	0.99783	0.99693
9:1	0.99700	0.99615	0.99678	0.99608	0.99787	0.99659	0.99682	0.99637	0.99764	0.99659

Table 5. F1-scores of the five algorithms evaluated on the PhiUSIIL dataset

Training–testing	ANN		CNN		LSTM		GRU		BiGRU	
ratio	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
5:5	0.99861	0.99811	0.99843	0.99828	0.99826	0.99814	0.99854	0.99812	0.99861	0.99827
6:4	0.99852	0.99821	0.99822	0.99814	0.99874	0.99832	0.99857	0.99834	0.99862	0.99842
7:3	0.99855	0.99817	0.99834	0.99827	0.99897	0.99842	0.99856	0.99843	0.99862	0.99833
8:2	0.99846	0.99802	0.99829	0.99803	0.99874	0.99824	0.99859	0.99822	0.99870	0.99827
9:1	0.99849	0.99800	0.99834	0.99800	0.99882	0.99811	0.99841	0.99818	0.99879	0.99822

Table 6 shows that BiGRU achieves the lowest test loss at the 6:4 training–testing ratio, with a value of 0.01114, indicating superior generalization capability. At the same ratio, LSTM records the lowest training loss of 0.00761; however, its test loss reaches 0.01235, which is slightly higher than that of BiGRU. In contrast, CNN consistently produces higher test losses across all configurations—for example, 0.01427 at the 6:4 ratio—reflecting poorer learning performance compared to the other models.

Table 6. Loss of the five algorithms evaluated on the PhiUSIIL dataset

Training–testing	ANN		CNN		LSTM		GRU		BiGRU	
ratio	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
5:5	0.01026	0.01387	0.01110	0.01296	0.01092	0.01260	0.01023	0.01298	0.00858	0.01285
6:4	0.01074	0.01316	0.01228	0.01427	0.00761	0.01235	0.00942	0.01166	0.00984	0.01114
7:3	0.01054	0.01319	0.01200	0.01366	0.00552	0.01255	0.00940	0.01166	0.01102	0.01293
8:2	0.01148	0.01535	0.01253	0.01417	0.00785	0.01267	0.00917	0.01221	0.00763	0.01294
9:1	0.01086	0.01406	0.01200	0.01440	0.00772	0.01420	0.01121	0.01437	0.00750	0.01359

In Figure 5, at the 6:4 training–testing ratio, the ANN model achieves the best classification performance, misclassifying only 1 phishing URL and 192 benign URLs. The BiGRU model follows closely, with 8 phishing and 162 benign misclassifications, demonstrating strong discriminatory capability between the two classes. In contrast, the LSTM and GRU models show weaker performance, misclassifying 16 phishing and 165 benign URLs, and 12 phishing and 167 benign URLs, respectively.

4.3. Discussion

Although phishing attacks via deceptive URLs are not a new cybersecurity issue, their increasing sophistication—particularly in financial transactions and online payment platforms—renders detection significantly more challenging, necessitating the development of intelligent and real-time adaptive solutions.

Accordingly, this study evaluates five DL models—ANN, CNN, LSTM, GRU, and BiGRU —using the PhiUSIIL dataset [20]. The dataset contains 235,795 URLs, including 134,850 benign and 100,945 phishing samples, and is divided into multiple training–testing ratios to identify the optimal split that maximizes classification accuracy while minimizing misclassification. This experimental design provides a consistent basis for comparing model effectiveness and identifying the most reliable architecture for phishing URL detection.

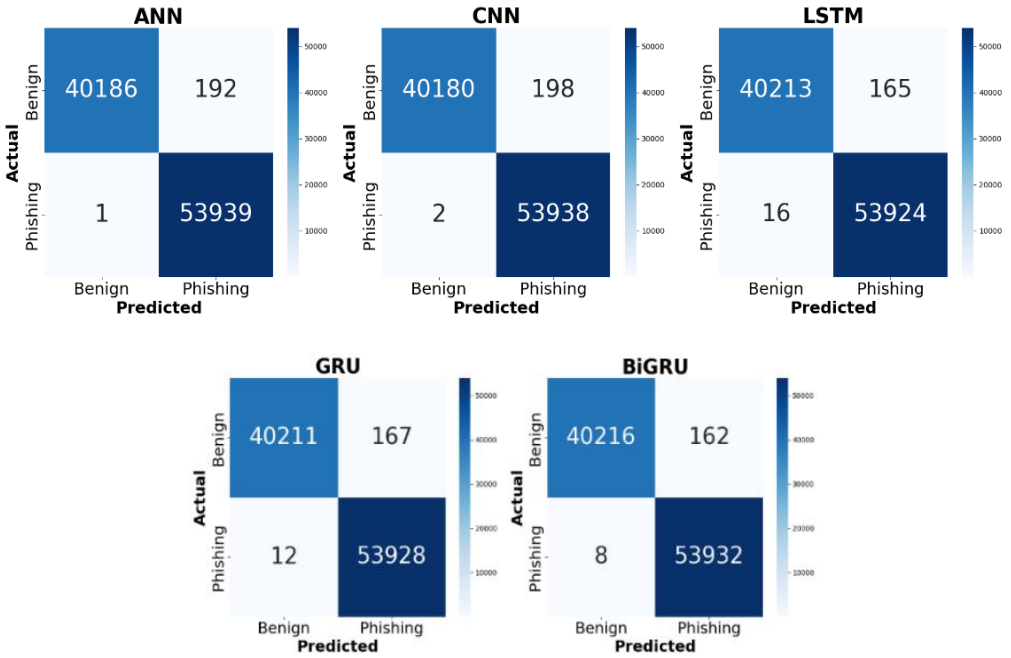

Figure 5. Misclassifications by the models at the 6:4 training–testing ratio

To ensure fairness and reproducibility, all models are implemented with the same configuration, including standardized inputs, architectures, and outputs (Table 1). Model performance is compared using multiple metrics: confusion matrix, accuracy, precision, F1-score, and loss. At the 6:4 split, BiGRU achieves the highest testing accuracy of 99.82%, surpassing GRU (99.81%), LSTM (99.80%), ANN (99.79%), and CNN (99.787%). BiGRU also maintains a stable accuracy trend above 99.8% across a wide range of splits (5:5 to 8:2), whereas the other models exhibit less consistent performance. Moreover, BiGRU outperforms its counterparts on other metrics at the 6:4 split, attaining a precision of 99.7% and an F1-score of 99.842%. Notably, it also achieves the lowest test loss of 0.01114, confirming its robustness and predictive efficiency. Therefore, BiGRU is recommended as the most reliable model for phishing URL classification.

For strengthening security and improving user convenience in real-time environments, particularly for financial transactions, a browser extension is developed in JavaScript and integrated into Chrome. This tool expands the character set from 270 to 366 symbols, enhancing the model's ability to detect unseen phishing URLs. The extension integrates the Ratcliff–Obershelp algorithm for fuzzy matching against two datasets—approximately 3,000 benign and 2,500 phishing URLs—stored in Cloud Firestore, with communication handled via REST API to ensure efficient real-time evaluation. The URL inspection process retrieves whitelist and blacklist data from Firestore, performs exact matching, and applies similarity scoring against stored entries with a predefined threshold. If the score is inconclusive, the pre-trained BiGRU model assigns the final label.

5. CONCLUSION

This study presents a comprehensive evaluation of DL models for phishing URL detection, identifying the BiGRU as the most effective architecture in terms of accuracy, precision, F1-score, and test loss across multiple training–testing ratios. BiGRU consistently achieved stable and superior results, with peak testing accuracy of 99.82% at the 6:4 split. To strengthen real-time phishing detection, the proposed framework integrates BiGRU into a lightweight Chrome extension, augmented with a 366-character encoding scheme and the Ratcliff–Obershelp similarity algorithm. This hybrid design combines similarity-based filtering with DL classification to detect even previously unseen URLs. Overall, the framework delivers a robust, efficient, and scalable solution for mitigating phishing threats in online environments.

The current study is limited to classifying URLs into phishing and benign categories, without addressing other attack vectors such as malware or scams. Future work will expand the dataset to include more diverse labels and investigate advanced models such as Transformer, BERT, and federated learning.

Furthermore, data augmentation techniques will be refined for pretraining to mitigate class imbalance, thereby improving the accuracy and reliability of real-time URL classification.

ACKNOWLEDGMENTS

The authors sincerely thank the Editor-in-Chief, the reviewers, and the Associate Editor for their constructive and valuable feedback. This research was financially supported by the Posts and Telecommunications Institute of Technology, under the Ministry of Information and Communications of Vietnam. Additional support was also provided by other external funding sources.

FUNDING INFORMATION

This paper is supported by Posts and Telecommunications Institute of Technology under grant number 10-2025-HV-CNTT2.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Dam Minh Linh	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓			✓
Han Minh Chau		✓								✓				✓
Huynh Trong Thua		✓								✓				
Tran Cong Hung	✓	✓		✓	✓					✓		✓	✓	✓

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study have been previously published and are openly available at GitHub: https://github.com/MinhLinhEdu/ptit-url-guardian.

REFERENCES

[1] APWG, "Phishing Activity Trends Report, Q2 2024," Anti-Phishing Working Group, [Online]. Available: https://docs.apwg.org/reports/apwg_trends_report_q2_2024.pdf.
[2] E. Kovacs, "SecurityWeek," Securityweek, Feb. 29, 2024 [Online]. Available: https://www.securityweek.com/discount-retail-giant-pepco-loses-e15-million-to-cybercriminals/. (Accessed: Jan. 26, 2025).
[3] N. A. Azeez, S. Misra, I. A. Margaret, L. Fernandez-Sanz, and S. M. Abdulhamid, "Adopting automated whitelist approach for detecting phishing attacks," *Computers and Security*, vol. 108, p. 102328, 2021, doi: 10.1016/j.cose.2021.102328.
[4] S. E. Elgharbi, M. A. Yahia, and S. Ouchani, "Online Phishing Detection: A Heuristic-Based Machine Learning Framework," in *2024 13th Mediterranean Conference on Embedded Computing (MECO)*, IEEE, Jun. 2024, pp. 1–4, doi: 10.1109/MECO62516.2024.10577848.
[5] M. Sabahno and F. Safara, "ISHO: improved spotted hyena optimization algorithm for phishing website detection," *Multimedia Tools and Applications*, vol. 81, no. 24, pp. 34677–34696, 2022, doi: 10.1007/s11042-021-10678-6.
[6] I. Kara, M. Ok, and A. Ozaday, "Characteristics of Understanding URLs and Domain Names Features: The Detection of Phishing Websites with Machine Learning Methods," *IEEE Access*, vol. 10, pp. 124420–124428, 2022, doi: 10.1109/ACCESS.2022.3223111.
[7] A. Awasthi and N. Goel, "Phishing website prediction using base and ensemble classifier techniques with cross-validation," *Cybersecurity*, vol. 5, no. 1, pp. 1–23, 2022, doi: 10.1186/s42400-022-00126-9.
[8] M. Aljabri *et al.*, "An Assessment of Lexical, Network, and Content-Based Features for Detecting Malicious URLs Using Machine Learning and Deep Learning Models," *Computational Intelligence and Neuroscience*, pp. 1–14, Aug. 2022, doi: 10.1155/2022/3241216.

[9] H. Alqahtani *et al.*, "Evolutionary Algorithm with Deep Auto Encoder Network Based Website Phishing Detection and Classification," *Applied Sciences*, vol. 12, no. 15, pp. 1–16, Jul. 2022, doi: 10.3390/app12157441.
[10] M. Bahaghighat, M. Ghasemi, and F. Ozen, "A high-accuracy phishing website detection method based on machine learning," *Journal of Information Security and Applications*, vol. 77, p. 103553, Sep. 2023, doi: 10.1016/j.jisa.2023.103553.
[11] B. V. Pavani, D. Mahitha, and B. U. Maheswari, "Enhancing Online Safety: Phishing URL Detection Using Machine Learning and Explainable AI," in *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, IEEE, Jun. 2024, pp. 1–6, doi: 10.1109/ICCCNT61001.2024.10723976.
[12] L. R. Kalabarige, R. S. Rao, A. R. Pais, and L. A. Gabralla, "A Boosting-Based Hybrid Feature Selection and Multi-Layer Stacked Ensemble Learning Model to Detect Phishing Websites," *IEEE Access*, vol. 11, pp. 71180–71193, 2023, doi: 10.1109/ACCESS.2023.3293649.
[13] T. Asif *et al.*, "RPCP-PURI: A robust and precise computational predictor for Phishing Uniform Resource Identification," *Journal of Information Security and Applications*, vol. 89, p. 103953, Mar. 2025, doi: 10.1016/j.jisa.2024.103953.
[14] H. Alqahtani and A. Abu-Khadrah, "Enhance the accuracy of malicious uniform resource locator detection based on effective machine learning approach," *Bulletin of Electrical Engineering and Informatics*, vol. 13, no. 6, pp. 4422–4429, Dec. 2024, doi: 10.11591/eei.v13i6.7797.
[15] S. J. Buu and S. B. Cho, "A Transformer network calibrated with fuzzy logic for phishing URL detection," *Fuzzy Sets and Systems*, vol. 517, p. 109474, Oct. 2025, doi: 10.1016/j.fss.2025.109474.
[16] V. L. Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, "TRANCO: A Research-Oriented Top Sites Ranking Hardened Against Manipulation," *arXiv preprint*, 2019, doi: 10.14722/ndss.2019.23386.
[17] PhishTank, "phishtank," Cisco Talos Intelligence Group, [Online]. Available: https://phishtank.org/. (Accessed: Jan. 28, 2025).
[18] D. M. Linh, "ptit-url-guardian," Github, [Online]. Available: https://github.com/MinhLinhEdu/ptit-url-guardian. (Accessed: Feb. 16, 2025).
[19] D. M. Linh, H. D. Hung, H. M. Chau, Q. S. Vu, and T. N. Tran, "Real-time phishing detection using deep learning methods by extensions," *International Journal of Electrical and Computer Engineering*, vol. 14, no. 3, pp. 3021–3035, Jun. 2024, doi: 10.11591/ijece.v14i3.pp3021-3035.
[20] A. Prasad and S. Chandra, "PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning," *Computers and Security*, vol. 136, p. 103545, Jan. 2024, doi: 10.1016/j.cose.2023.103545.
[21] O. Rainio, J. Teuho, and R. Klén, "Evaluation metrics and statistical tests for machine learning," *Scientific Reports*, vol. 14, no. 1, pp. 1–14, 2024, doi: 10.1038/s41598-024-56706-x.
[22] Z. Huang, T. Ban, and Y. Zhang, "A novel approach for malicious URL detection using RoBERTa and sparse autoencoder," *Journal of Information Security and Applications*, vol. 94, p. 104214, Nov. 2025, doi: 10.1016/j.jisa.2025.104214.
[23] F. Rashid, B. Doyle, S. C. Han, and S. Seneviratne, "Phishing URL detection generalisation using Unsupervised Domain Adaptation," *Computer Networks*, vol. 245, pp. 1–14, May 2024, doi: 10.1016/j.comnet.2024.110398.
[24] C. Wang and Y. Chen, "TCURL: Exploring hybrid transformer and convolutional neural network on phishing URL detection," *Knowledge-Based Systems*, vol. 258, p. 109955, Dec. 2022, doi: 10.1016/j.knosys.2022.109955.
[25] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Cambridge, MA, USA: The MIT Press, 2016.
[26] M. Wang, L. Song, L. Li, Y. Zhu, and J. Li, "Phishing webpage detection based on global and local visual similarity," *Expert Systems with Applications*, vol. 252, p. 124120, Oct. 2024, doi: 10.1016/j.eswa.2024.124120.
[27] A. Connolly and H. F. Atlam, "Effective ensemble learning phishing detection system using hybrid feature selection," *Journal of Network and Computer Applications*, vol. 242, p. 104251, Oct. 2025, doi: 10.1016/j.jnca.2025.104251.

BIOGRAPHIES OF AUTHORS

Dam Minh Linh was born in 1982 in Tay Ninh province, Vietnam. He received his engineering degree in Information Technology in 2010 and his M.Sc. degree in Information Systems in 2016, both from the Posts and Telecommunications Institute of Technology. He is currently a lecturer at the Faculty of Information Technology 2 and a member of the Information Security Technology Lab at the Posts and Telecommunications Institute of Technology, Ho Chi Minh City, Vietnam. His research interests include anti-phishing solutions, cyber threat intelligence, deep learning with transformer models, natural language processing, information retrieval (IR), and computer vision. Since 2024, he has been a Ph.D. student in Information Systems at the Posts and Telecommunications Institute of Technology, Vietnam. His current research focuses on applying transformer-based deep learning techniques to combinatorial optimization, particularly in cybersecurity and network security domains. He can be contacted at email: linhdm@ptit.edu.vn.

Han Minh Chau was born in Vietnam in 1980. He is currently the Head of the Department of Computer Networks, Faculty of Information Technology, HUTECH University, Ho Chi Minh City, Vietnam. He received a Bachelor's degree in Information Technology from Hanoi University of Science and Technology and a Master's degree in Computer Science from the Ho Chi Minh City University of Information Technology, Vietnam National University at Ho Chi Minh City. His main research interests include network security, machine learning, computer vision, cloud computing platforms, and artificial intelligence. He can be contacted at email: hm.chau80@hutech.edu.vn.

Huynh Trong Thua is currently the head in the Department of Information Security, Faculty of Information Technology 2, at the Posts and Telecommunications Institute of Technology (PTIT), Vietnam. He received a bachelor's degree in information technology from Ho Chi Minh City University of Natural Sciences, a master's degree in computer engineering from Kyung Hee University, Korea, and a Ph.D. degree in computer science from the Ho Chi Minh City University of Technology, Vietnam National University at Ho Chi Minh City. His key areas of research include cybersecurity, AI and big data, and intelligent information systems. He can be contacted at email: thuaht@ptit.edu.vn.

Tran Cong Hung was born in Vietnam in 1961. He received a B.E. in Electronics and Telecommunication Engineering with first-class honors from Ho Chi Minh University of Technology in 1987. He received a B.E. in Informatics and Computer Engineering from Ho Chi Minh University of Technology in 1995. He earned a Master of Engineering degree in Telecommunications Engineering from the Postgraduate Department of Hanoi University of Technology in 1998. He received a Ph.D. from Hanoi University of Technology in 2004. His main research areas include B-ISDN performance parameters and measurement methods, QoS in high-speed networks, and MPLS. Currently, he is an associate professor at the Department of Computer Science and Engineering, Saigon International University in Ho Chi Minh City, Vietnam. He can be contacted at email: tranconghung@siu.edu.vn; https://tranconghung.com.