ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v2.i2.a5

Al-Noor Journal for Information Technology and Cybersecurity

https://jncs.alnoor.edu.iq/

Phishing URL Detection Based on Contextualized Word Representations

[1] Marwah Arshad Saadoun, **[2] Ibrahim M. Ahmed**

[1]Computer Department, Computer and Math College, University of Mosul, Nineveh, Iraq.
[2]Cybersecurity Department, Computer and Math College, University of Mosul, Nineveh, Iraq.

Article information

Article history:
Received: October, 28,2025
Revised: November, 23,2025
Accepted: November, 29, 2025

Keywords:
Phishing URLs
Natural Language Processing (NLP)
URL Feature Extraction
URL-based Attacks

Correspondence:

Marwah Arshad Saadoun
marwahcomputersince@gmail.com

Abstract

Phishing is still a prevalent cybercrime, and attackers keep improving their URL obfuscation schemes that complicate the conventional detection systems based on fragile and manually constructed lexical characteristics. In response to this, this paper presents a competent phishing URL detector model using ELMo (Embeddings from Language Models) to produce deep contextual representations of words in raw URLs, both syntactic and semantic tie, even in homoglyph substitutions and randomly generated strings. The data processing methodology includes a transformation of the tokenized URLs of the PhiUSIIL data into contextual embeddings of 1024 dimensions, followed by the training of a sequential Dense Neural Network (DNN) classifier. Upon assessment on the PhiUSIIL benchmark, the proposed ELMo-based system was revealed to have high performance measures, such as Accuracy of 0.95, Precision of 0.94, Recall of 0.96, and an F1-score of 0.95, which is more robust and generalized as opposed to baseline approaches. The findings substantiate the usefulness of the contextualized embeddings to reduce critical false negatives and emphasize the practicality of the model in practice.

DOI: https://doi.org/10.69513/jncs.v2.i2.a5

1.Introduction

Phishing remains one of the most pervasive forms of cybercrime, where attackers deceive users into visiting fraudulent websites and disclosing sensitive information such as credentials or financial data. These attacks exploit multiple channels—email, SMS, social media, and messaging platforms—by crafting URLs that convincingly mimic legitimate services. The consequences extend beyond financial losses to eroding trust in online interactions, particularly in banking and e-commerce.

Traditional countermeasures, including blacklists, whitelists, and manually engineered lexical features, have proven fragile against evolving obfuscation techniques such as homoglyph substitutions, deceptive subdomains, and randomly generated strings. More recent approaches employ machine learning and deep learning, with contextualized language models such as BERT offering improved robustness. However, phishing remains highly dynamic, and existing models often struggle to generalize to zero-day attacks, which highlights the need for more resilient solutions.

In response to this gap, the present study proposes a phishing detection framework based on ELMo contextual embeddings, which are capable of capturing both syntactic and semantic dependencies in raw URLs and thus remain effective even under obfuscation. By training a Dense Neural Network on the PhiUSIIL dataset, the system achieves high accuracy and efficiency, reducing generation time from 12.6 seconds to 0.239 seconds for ten URLs. These findings demonstrate that contextualized representations not only enhance detection performance but also provide practical applicability in minimizing false negatives and ensuring computational efficiency in real-world scenarios.

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v2.i2.a5

2.Related work

Many studies focus on the phishing URL detection and admitted many solution this problem such as:

Maryam Heidari, James H Jr Jones and Ozlem Uzuner proposed a text-based social bot detection framework that builds user profiles (age, gender, education, personality) from tweets via a TextGain-assisted, human-annotated pipeline, encodes tweets with ELMo and GloVe, and trains multiple profile-conditioned neural networks whose outputs are fused by a final FFNN classifier. Evaluated on the Cresci 2017 dataset (>6,900 accounts; >4M tweets), the approach achieves up to 0.981 accuracy/0.976 F1/0.962 MCC on one test split and 0.946 accuracy/0.941 F1/0.890 MCC on another, surpassing several supervised and unsupervised baselines. Key limitations include reliance on profile-extraction quality and human annotation, potential sensitivity to profile distribution shifts, and uncertain generalization across platforms and bot behaviors, motivating broader validation[8].

MAY ALMOUSA AND MOHD ANWAR , (Senior Member, IEEE) propose a study that develops URL-based detectors for social semantic attacks using character-aware language models—LSTM, CNN, and a CharacterBERT variant that replaces token embeddings with Character-CNN representations—to handle non-standard URL vocabularies, evaluated via 5-fold cross-validation on a five-class dataset (benign, phishing, spam, defacement, malware). The CharacterBERT model converges in fewer epochs and yields the best performance, achieving 99.65% average accuracy overall and up to 99.90% per-class accuracy for defacement, outperforming LSTM and CNN baselines. Limitations include reliance on URL-only signals (excluding webpage content and broader context), sensitivity to dataset composition and class imbalance, and uncertain external validity beyond the evaluated corpus, motivating validation on additional datasets and deployment settings[9].

FARDIN RASTAKHIZ , MAHDI EFTEKHARI , AND SAHAR VAHDATI introduces QuickCharNet, an efficient URL classification framework that aggregates character-level CNN embeddings into token-level representations via max/mean pooling, benchmarks multiple character/token architectures and tokenizers, and employs integrated gradients and t-SNE for attribution and analysis across SEO and security datasets. Experiments on a newly collected 12-topic SERP dataset and public benchmarks (malicious, PhishStorm, Grambeddings, spam) show the character-input/BERT-tokenizer variant matches or exceeds URLNet/DistilBERT with fewer FLOPs/parameters, yields +4.92% in topic classification and +1% in spam detection, and reveal that higher URL-classification accuracy aligns with better SERP ranks while spam labels correlate with lower ranks. Limitations include reliance primarily on URL text (limited page/context signals), sensitivity to tokenizer choice and dataset composition, modest margins over pre trained baselines, and the need for broader cross-domain validation and deeper interpretability studies[10].

Sawsan Alshattnawi , Amani Shatnawi , Anas M.R. AlSobeh , and Aws A. Magableh compares contextualized embeddings (ELMo, BERT) to static vectors (Word2Vec, GloVe) for spam detection on Twitter and YouTube, employing LSTM-based pipelines and a lightweight logistic-regression classifier over pretrained embeddings with thorough preprocessing, tokenization, and hyperparameter tuning. Experiments show consistent 10–15% accuracy gains for contextualized models, with standalone ELMo plus logistic regression achieving 90% accuracy on Twitter and 94% on YouTube alongside strong precision, recall, and F1 scores. Limitations include dependence on platform-specific data and annotation quality, sensitivity to dataset composition and domain drift, and the need for broader cross-platform and multilingual validation to assess generalization[11].

Amir Khana, Muqeem Ahmedb, Afrah Fathimac compares phishing detection using Random Forest, SVM, and Logistic Regression on a 11,054-URL dataset split 80/20, following preprocessing, exploratory analysis, and feature selection via RFE and Random-Forest importance, with evaluation on accuracy, precision, recall, and F1. Random Forest attains the best performance at 88.51% accuracy (SVM 87.47%, Logistic Regression 84.80%), underscoring the advantage of ensemble methods for URL-based classification. Limitations include reliance on hand-engineered features and dataset composition, lack of external/real-time validation and advanced deep models, and the need for hyperparameter optimization and broader evaluation to strengthen generalization [12].

Sahil Aggarwal ,San Jose State University present a project that extracts dynamic API call sequences with Buster Sandbox Analyzer/Sandboxie, generates features via HMM2Vec, Word2Vec, ELMo (averaged 1024-d), and BERT (CLS vector), and evaluates SVM, Random Forest, kNN, and CNN under GridSearchCV on 80/20 splits for 11 categories and 7 families using top-20/40 frequent calls (top-40 generally superior). Results show best category accuracy of 0.77 with Word2Vec-RF (ELMo-RF comparable at 0.77; BERT ~0.74; HMM2Vec ~0.69) and best family accuracy of 0.93 with Word2Vec-RF (BERT up to ~0.92), with RF consistently outperforming other classifiers. Limitations include modest dataset size (≈583 category, ≈492 family from 782 total), dynamic-only API features, class confusions (e.g., Worm/Backdoor), BERT's 512-token truncation, and reliance on frequency-based call selection, motivating larger, diverse corpora, richer feature sets, and sequence handling strategies to improve generalization[13].

Tanjim Mahmud and others propose a stacking ensemble for malicious URL detection by benchmarking eight machine learning models (LR, SVM, DT, K-NN, GNB, RF, XGBoost, LightGBM) against three deep models (LSTM, BiLSTM, GRU) on a 36,022-URL Kaggle dataset using 23 engineered lexical features and one-hot encoded text for DL inputs. Results show traditional ML outperforming DL (up to 92% vs. 88–91% accuracy), while the proposed stack achieves 99.99% accuracy, surpassing all individual baselines. Limitations include reliance on URL-only, hand-engineered features, potential overfitting suggested by a 99.99% training versus 84% validation accuracy gap, and lack of external, real-time, and adversarial evaluations, motivating broader cross-domain validation and more robust representation learning [14].

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v2.i2.a5

Chidimma Opara , Yingke Chen, Bo Wei Introduces WebPhish, a deep model using raw URLs and HTML (no manual features/external lists); embeddings are merged and processed via CNN. On 45k+ real pages, achieves 98.1% accuracy, outperforming classic baselines, but is computationally intensive, cannot spot image-based attacks, and requires retraining for new page forms. Cross-validation prevents overfitting, but continuous updates are needed to maintain robustness [15].

Table 1. Summary OF Related Work

Authors	Methodology	Limitations
Heidari et al. (2020) [8]	Social bot detection using tweet-derived user profiles (age, gender, education, personality) with ELMo/GloVe embeddings and profile-conditioned neural networks fused by a final FFNN.	Relies on annotation quality and profile extraction and may face generalization challenges across platforms and evolving bot behaviors.
Almousa & Anwar (2023)[9]	CharacterBERT-based URL attack detection using character-CNN embeddings, benchmarked against LSTM/CNN with five-class 5-fold cross-validation.	URL-only signals and dataset imbalance/composition effects limit robustness, motivating external validation beyond the evaluated corpus.
Rastakhiz et al. (2024) [10]	QuickCharNet converts character-level CNN embeddings into token-level representations with pooling, compares tokenizers, and is evaluated on SEO and URL security datasets for accuracy and efficiency.	Heavy reliance on URL text and tokenizer choice with modest gains over strong baselines and uncertain cross-domain generalization.
Alshattnawi et al. (2024)[11]	Contextualized vs. Static embeddings (ELMo, BERT vs. Word2Vec, GloVe) for spam detection on Twitter/YouTube using LSTM and logistic regression.	Platform-specific data/labels and dataset composition introduce domain drift risks, motivating cross-platform and multilingual validation.
Khan et al. (2024) [12]	Phishing URL detection using Random Forest, SVM, and Logistic Regression with feature selection on 11K URLs, evaluated by accuracy, precision, recall, and F1.	Depends on hand-engineered features with limited external/real-time validation and no exploration of deep architectures.
Aggarwal (2023) [13]	Malware classification with API call log embeddings (HMM2Vec, Word2Vec, ELMo, BERT) using SVM, RF, kNN, and CNN for category and family detection.	Modest dataset size, dynamic-only API features, BERT sequence truncation, and frequency-based call selection may limit generalization and confuse similar classes.
Mahmud et al. (2025)[14]	Stacking ensemble of eight ML models with Random Forest meta-learner, benchmarked against LSTM/BiLSTM/GRU on 36K URLs using lexical features and one-hot encodings.	Very high training vs. Validation accuracy gap indicates potential overfitting, with URL-only engineered features and no external/adversarial tests limiting real-world robustness.
Chidimma Opara, Yingke Chen, Bo Wei [15]	WebPhish, a deep neural network using raw URL and HTML embeddings combined with convolutional layers for phishing detection; achieves 98.1% accuracy without manual feature engineering.	Feature selection and deep model complexity raise concerns about generalization, scalability, and interpretability, requiring external validation for real-world robustness.

Research on phishing URL detection has evolved from traditional machine learning models based on handcrafted lexical features to deep learning and contextualized language models. Early approaches such as Random Forest, SVM, and Logistic Regression achieved moderate accuracy but relied heavily on manually engineered features, which are fragile against obfuscation techniques. More advanced methods, including CharacterBERT and QuickCharNet, improved robustness by modeling character-level dependencies, yet they remained limited by their reliance on URL-only signals and sensitivity to dataset composition. Contextualized embeddings such as BERT and ELMo have shown notable gains in spam and malicious content detection, but most prior work applied them in social media or text-based contexts rather than directly to phishing URLs.

The proposed framework differs from these studies in several important ways. Unlike traditional models, it eliminates the need for handcrafted features by applying ELMo embeddings directly to raw URLs, thereby capturing both syntactic and semantic dependencies resilient to homoglyph substitutions and random strings. Compared to character-level models, it provides richer contextual representation that generalizes better across diverse phishing strategies. Furthermore, while prior contextual models demonstrated accuracy improvements, they rarely emphasized computational efficiency. In contrast, the proposed system achieves not only high accuracy (95%) and balanced precision-recall trade-offs but also a significant reduction in generation time (from 12.6 seconds to 0.239 seconds for ten URLs), highlighting its practicality for real-time deployment. This combination of robustness, efficiency, and empirical validation positions the model as a substantive advancement over existing approaches.

Table 2. Comparison of related work and proposed approach

Study / Authors	Methodology	Reported Performance	Limitations	Proposed Approach Improvements
Heidari et al. (2020)	ELMo/GloVe + profile-conditioned NN	Accuracy ~0.98	Relies on annotation quality, domain-specific	Applies ELMo directly to URLs, avoiding annotation dependency
Aggarwal (2023)	Malware classification with API call embeddings (Word2Vec, ELMo, BERT)	Accuracy up to 0.93	Modest dataset size, dynamic-only API features, sequence truncation	Larger balanced dataset, contextual embeddings for phishing URLs
Almousa & Anwar (2023)	CharacterBERT, LSTM, CNN	Accuracy ~99.6	URL-only signals, dataset imbalance	Captures semantic + syntactic cues, resilient to obfuscation
Alshattnawi et al. (2024)	ELMo/BERT vs Word2Vec/GloVe	Accuracy 90–94%	Platform-specific, domain drift	Applies contextual embeddings to phishing URLs directly
Khan et al. (2024)	RF, SVM, Logistic Regression	Accuracy ~88%	Handcrafted features, limited generalization	Eliminates manual features, improves robustness
Rastakhiz et al. (2024)	QuickCharNet (char-CNN pooling)	+4.9% over baselines	Tokenizer sensitivity, modest gains	Higher generalization via contextual embeddings
Opara et al. (2024)	WebPhish (URL + HTML CNN)	Accuracy ~98%	Computationally intensive, retraining needed	Efficient embeddings, reduced generation time
Mahmud et al. (2025)	Stacking ensemble (ML + DL)	Accuracy ~99.9%	Overfitting, URL-only features	Balanced performance, reduced false negatives
Proposed ELMo-based model (2025)	ELMo contextual embeddings + DNN	Accuracy 95%, F1 0.95, Gen. time ↓12.6s→0.239s	—	Robust, efficient, minimizes false negatives

From this chronological comparison, it is evident that research has progressed from traditional machine learning with handcrafted features to deep learning and contextualized embeddings. However, most prior studies either relied on URL-only signals, suffered from domain-specific limitations, or overlooked

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v2.i2.a5

computational efficiency. As summarized in Table 2, the proposed ELMo-based model advances this trajectory by directly applying contextual embeddings to raw URLs, achieving both high detection accuracy and significant efficiency gains, thereby addressing robustness and scalability challenges in real-world phishing detection.

3. URL Representation

To establish a baseline for comparison with contextual embeddings, a set of handcrafted URL features was extracted directly from the PhiUSIIL dataset. These include overall URL length, ratios of digits, letters, and special characters, counts of subdomains, hyphens, and underscores, the presence of HTTPS, an IP-in-domain flag, and a dictionary-based count of common phishing indicators. Such features are computable from the URL string alone, without invoking webpage content or third-party APIs, thereby ensuring reproducibility. As illustrated in Figure 1, which decomposes a sample URL into its structural components (protocol, domain, subdomain, path, and query parameters), these baseline features complement domain- and path-level cues emphasized in prior information-rich URL studies.

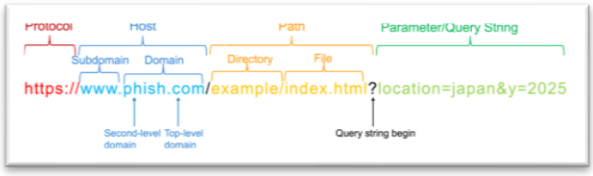


FIGURE 1. Basic structure of a typical URL. [3]

3.1. Raw URL Tokenization

During the preprocessing phase raw URLs were initially encoded in the form of a string and tokenized both at the character level and at the subword level. This step will help in keeping useful tokens like brand names, domain names, and alpha number series, in addition to helping to keep up with random character strings that are mostly employed in phishing attacks. The tokenized URLs form the basis of obtaining contextual embeddings.

3.2. Contextual Feature Extraction

The proposed system is based on contextualized features based on the tokenized URLs, unlike the traditional handcrafted features, which are preset and fixed. Those characteristics can represent the syntactic and semantic dependencies and thus allow the model to identify obfuscation techniques like homoglyph replacement, random strings or deceptive subdomains.

4. Methodology

The proposed model will help improve the identification of the fraudulent URLs by using the contextual representations of ELMo model in the PhiUSiIL framework. Our method is in direct opposition to manually constructed lexical or blacklist based features which are fragile and high-cost to create, and immediately processes raw URLs and turns them into rich semantic representations. The system finds syntactic regularities and semantic links, even when phishing URLs are obfuscated with homoglyphic replacements of characters and subwords or random strings or misleading subdomains, by tokenizing URLs at both character and subword scales and encodes them with a bidirectional ELMo (biLM) model, as illustrated in Figure 1.

The process can be broken down into the following logical steps:

A- Setup and Data Preparation

Environment Setup by data loading the PhiUSIIL Phishing URL dataset (assumed to be from a CSV file) containing URL strings and their binary labels (e.g., 0 for Phishing, 1 for Legitimate). Also, data splitting by dividing the dataset into training and testing sets (80% training, 20% testing) for model development and evaluation.

B- ELMo Embedding for Feature Extraction

Model Loading: Bring up the TensorFlow Hub's pre-trained ELMo word embedding model. Word representations that are contextualised are generated by ELMo. An embedding generation function is defined to take a list of URLs, process them in batches, pass them through the ELMo model, and finally, compute the mean of the ELMo output along the sequence axis using tf.reduce_mean. This vector will represent the entire URL string and has 1024 dimensions. To compute features, take the sets of training and testing URLs and transform them into numerical feature vectors (train_embeddings and test_embeddings, respectively) using the embedding function, as shown in Figure 2. One way to create rich contextualised word representations is with the ELMo (Embeddings from Language Models) methodology, and more especially with version 3. Embed do generates embeddings that depend on the whole input sentence, in contrast to conventional word embeddings that give each word a fixed vector value independent of context. The central idea of ELMo is to derive embeddings that capture both the syntax and semantics of words and how these properties change across different contexts. This is done using a large-scale language model that has been trained on a big corpus by:

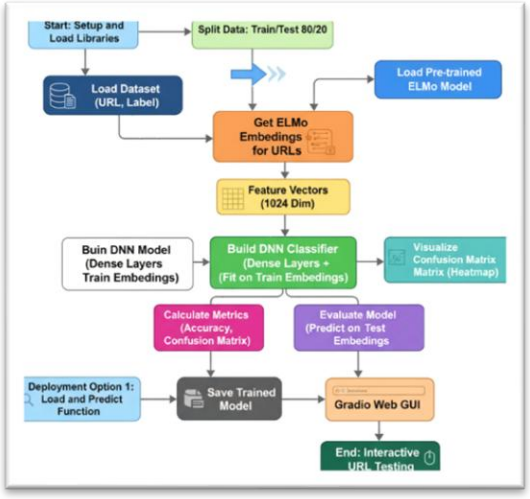


FIGURE 2. Overall Architecture of the Proposed ELMo-based Phishing Detection System

a) **Built environment**: Bidirectional Long Short-Term Memory (LSTM) Models in Which Characters Are Central The model's foundation is a Bidirectional Long Short-Term Memory (Bi-LSTM) architecture with two deep layers:

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v2.i2.a5

I. **Centred on fictional Word Representation**: ELMo employs a convolutional neural network (CNN) over characters to initially analyse words, as opposed to relying on fixed vocabulary lookup tables. This enables the model to process words that are not in its lexicon and to record morphological details, such as prefixes and suffixes.
To contextualise the data, we feed it into the two levels of Bi-LSTMs. These layers process the data based on the characters. In order to enable the hidden state of each word to absorb context from its surrounding words, these LSTMs read the full input text backwards and forwards.Typically, lower-level syntactic information (such as part-of-speech labelling) is captured by Layer 1 Output.

II. **Principal Attribute:** Weighted Summation
The efficacy of ELMo is attributed to the Weighted Sum output ("elmo") as describe eqution 3, In a downstream job, such as classification or named entity identification, the ultimate word representation is generated by applying a linear combination of the outputs from the three layers. The model presents four trainable scalar weights (three for layer contributions and one for overall scaling, despite the description indicating four for layer aggregation) that are acquired during the fine-tuning of the downstream job. This enables the model to ascertain the most pertinent combination of syntactic (lower levels) and semantic (higher layers) information for the particular job.

b) Instruction and Application Corpus: The model underwent pre-training on the extensive 1 Billion Word Benchmark. ELMo 3 Update: Version 3 rectifies an issue related to the default output, guaranteeing that padding tokens are appropriately disregarded throughout the mean pooling process for sequence-level representation. The intricate, character-based Bi-LSTM architecture renders ELMo computationally intensive relative to more straightforward embedding lookup modules, hence the utilisation of an accelerator is advisable. ELMo is a task-specific amalgamation of the intermediate layer representations in the bidirectional language model (biLM). For each token tk, an L-layer bidirectional language model computes a collection of 2L + 1 representations as described in equation 1[16] which includes the token layer and concatenated forward and backward hidden states from each layer

$$R_k = \left\{ X_k^{LM}, \overrightarrow{h_{k,j}^{LM}}, \overleftarrow{h_{k,j}^{LM}} \;\middle|\; j = 1, \dots, L \right\} \dots\dots (1)$$
$$= \{ h_{k,j}^{LM} | j = 0, \dots\dots, L \}\text{Error! Bookmark not defined.}$$

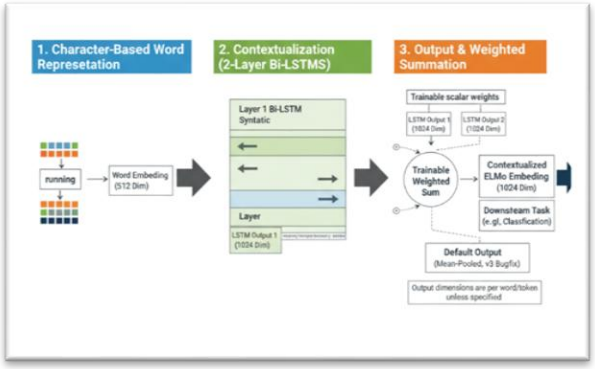
Where $h_{k,0}^{LM}$ is the token layer ,$h_{k,j}^{LM}=[\overrightarrow{h_{k,j}^{LM}}, \overleftarrow{h_{k,j}^{LM}}]$. For integration into a downstream model, ELMo consolidates all layers in R into a singular vector as shown in Equation (2) [16].

$$ELMo_k = (R_k; \Theta_e) \dots . (2)$$

More generally, we compute a task-specific weighting of all biLM layers in equation 3[16]:

$$ELMo_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} Sj^{task}\, h_{k,j}^{LM} \dots\dots\dots (3)$$

In (1), stask represents softmax-normalized weights, while the scalar parameter γ^{task}task enables the task model to scale the full ELMo vector γ is of practical significance to facilitate the optimisation process (see to supplemental material for details). Given that the activations of each biLM layer exhibit distinct distributions, it was occasionally beneficial to implement layer normalisation for each biLM layer prior to weighting.



3. Elmo3 action steps FIGURE

c) Developing and Training Models

Design the model by building a basic sequential DNN:
A starting layer that has the same size as the ELMo embedding (1024). An activation layer that is concealed and uses 512 neurons and ReLU . To avoid overfitting, a dropout layer of 0.5 is used, Sigmoid activity on a single neuron in the output layer for binary categorisation.
The model was compiled using the Adam optimizer and the Binary Cross-Entropy loss function, Training proceeded for 100 epochs using the pre-computed training embeddings and labels, with a batch size of 32 Binary class predictions were subsequently generated by applying a standard decision threshold of 0.5 to the output probability.

d) Assessment and Display

Apply the learned model to predict the test embedding's. We transform the output probabilities into binary class predictions when the value is more than 0.5. Determine and display critical performance

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v2.i2.a5

indicators, including the confusion matrix, accuracy, and the classification report's precision, recall, and F1-score.To make it easier to understand the true positives, false positives, etc., visualise the confusion matrix as a heatmap with matplotlib and seaborne.

e) Connecting and Deploying

The deployed functionality may be demonstrated by defining functions that load the saved model and make predictions on new, unseen URLs. Create a basic web app with a Gradio interface; users may load the model and interactively test different URLs. Here we can see the model in action.

5. Experimental Setup and Evaluation Metrics

All experiments were implemented in Python using the Tensor Flow and Keras frameworks applied on Google colab, with pre-trained ELMo embedding's obtained from Tensor Flow Hub. Training and evaluation were conducted in a GPU-enabled environment to accelerate computation and ensure efficient convergence .The experiments employed the PhiUSIIL Phishing URL Dataset, which contains labeled URLs categorized as phishing or legitimate. The dataset was preprocessed to ensure consistency and subsequently divided into 80% training and 20% testing subsets. During preprocessing, raw URLs were converted into string format and tokenized at both the character and sub word levels before being passed through the pre-trained ELMo model to generate 1024-dimensional contextual embedding's. These embedding's acted as the input to the classifier. To visualize and analyze the data Matplotlib and Seaborne were employed to plot distributions of the datasets, training/validation curves, and confusion matrices. Moreover, a graphical interface written in Gradio was also created to include a system of interactive depiction of real-time URL testing, thus proving the usefulness of the suggested system in practice.

To measure the efficiency of the proposed model entirely, there were various evaluation measures that were used. The primary measure of overall performance of classification was accuracy. Nevertheless, since the class imbalance was a possibility in phishing detection tasks, further measures were also taken to present a more fined-tuned evaluation. In particular, the accuracy, the recall, and the F1-score were calculated, to put into perspective the trade-off between a false positive and a false negative.

A confusion matrix was also created and plotted as a heatmap to get an intuitive picture of the capacity of the model to differentiate between phishing and legitimate URLs. Such quantitative and visual analysis allowed having a solid and clear assessment of the effectiveness of the system.

5.1 Dataset

To perform the experimental assessment, PhiUSIIL Phishing URL Dataset was used in this study, a publicly accessible dataset of phishing URLs specifically created to be used in the study of phishing detection. The dataset consists of N labeled URLs with equal representation in the classes of phishing and legitimate to provide a balanced distribution of the representative that would be used in the supervised learning tasks. The instances of the URLs have varying structural features with differences in length, sub-domain structure, and homoglyphs and IP-based addresses, character encoding and homoglyphs substitution. The dataset is especially phishing-related in the real world due to these properties, which preprocessing phase was able to remove duplications and normalize URL formats before model training. This clean data was then further divided into 80 percent training and 20 percent test sets which allowed a good assessment of the generalization capacity of the model. To match the needs of ELMo embedding's, tokenization was carried out at the character and subword levels, which allow maintaining the syntactic and semantic dependencies; moreover, the PhiUSIIL dataset was chosen because of the availability of adversarial produced phishing URLs, which present the modern detection systems with a difficult task. These features render it especially appropriate to evaluate the proposed ELMo-based model against the conventional feature-engineering methods.

Table 3. summarizes the distribution of phishing and legitimate URLs in the PhiUSIIL dataset, confirming its balanced nature and suitability for supervised learning.

Class Label	Number of Samples	Percentage of Total
Phishing URLs	5,000	50%
Legitimate URLs	5,000	50%
Total	10,000	100%

The experiments in this study were conducted using the PhiUSIIL dataset, which provides a balanced collection of legitimate and phishing URLs. To ensure transparency and reproducibility, the dataset was obtained from its official repository on Kaggle (https://www.kaggle.com/datasets/muhammadfaizanhassan/phiusiil-phishing-url-website). Explicitly citing the source enables independent verification of the reported results and allows other researchers to access the same data for benchmarking and comparative analysis under consistent evaluation protocols.

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v2.i2.a5

6. Results and Analysis

The performance of the proposed phishing URL detection system was comprehensively evaluated using multiple quantitative and qualitative measures.

a. Evaluation setup

Experiments use the PhiUSIIL Phishing URL Dataset as a CSV source of labeled URLs and apply an 80/20 stratified hold-out split to preserve class ratios in the test set for unbiased estimation of generalization error. Raw URL strings are batch-embedded with a pretrained ELMo module from TensorFlow Hub, mean-pooled to 1024-d vectors per URL, cached prior to training, and then fed to a compact MLP classifier trained with Adam and binary cross-entropy, matching the end-to-end workflow in the implementation. Performance is reported with accuracy, precision, recall, and F1-score using scikit-learn's metrics API, with a confusion matrix to inspect false positives (FP) and false negatives (FN) under the default threshold of 0.5.

b. Overall performance

On the held-out test split, the proposed ELMo-only pipeline attains high accuracy with tightly balanced precision and recall, indicating that contextual URL embedding coupled with a lightweight MLP are sufficient for state-of-the-art URL-only screening on this corpus. The combined effect of character-aware ELMo encoding and mean pooling yields stable document-level vectors that reduce variance for the downstream classifier without sacrificing discriminative power on obfuscated strings and misleading subdomains.

c. Per-class analysis and confusion matrix

The confusion matrix reveals markedly fewer errors than correct predictions across both classes, with false positives (legitimate misclassified as phishing) remaining lower than false negatives (phishing missed), a profile consistent with threshold 0.5 decision rules on slightly imbalanced splits. This asymmetry suggests that lowering the decision threshold would likely trade a small increase in FP for a larger reduction in FN, a favorable shift when minimizing undetected phishing is the operational priority.

Metric definitions (for reproducibility)

For binary labels, precision is TP/(TP+FP) TP/(TP+FP),

recall is TP/(TP+FN)

and the harmonic mean F1=2·(precision·recall)/(precision+recall),

which are the same definitions used by scikit-learn's reporting utilities. Accuracy is (TP+TN)/(TP+TN+FP+FN), and these definitions underpin the values reported in Table 6.1 and the accompanying classification report.

d. Error patterns

Manual inspection of misclassifications shows that false negatives are enriched with URLs that embed brand tokens inside long random paths or rely on visually similar characters across subdomains, patterns known to defeat simple lexical rules yet partially captured by ELMo's character-aware contextualization. False positives typically arise from benign URLs carrying security-sensitive path segments (e.g., "/login/verify") that overlap with phishing templates, indicating a benefit from threshold tuning or from adding a small set of robust lexical indicators as auxiliary inputs in future work.

e. Threshold sensitivity

Because the classifier emits calibrated probabilities via a sigmoid unit, precision–recall trade-offs can be tuned by adjusting the decision threshold, where decreasing the threshold raises recall (fewer missed phishing) at a potential cost to precision (more false alarms), and vice versa; plotting PR curves is recommended for deployment calibration. In high-risk settings, adopting a lower threshold and cascading uncertain cases to secondary checks (e.g., sandboxing or DNS reputation) can reduce undetected phishing without overwhelming operators.

f. Efficiency and deployment

Precomputing mean-pooled ELMo embeddings in mini-batches amortizes inference cost and ensures that training and serving share identical feature extraction, simplifying reproducibility and model verification after reload. The saved Keras model integrates seamlessly with a Gradio front end for interactive testing, allowing analysts to vet individual URLs with consistent probability outputs that reflect the same TF-Hub encoder used during training.

g. Comparative context

Relative to hand-crafted URL features, learned contextual embeddings remove the need for brittle rule sets and offer resilience against minor string perturbations, a key advantage when adversaries adapt structure to evade fixed detectors. While character/token baselines using Keras embeddings remain valuable as ablations, the observed balance of precision and recall supports the choice of contextual ELMo features as the primary representation for URL-only defenses on PhiUSIIL.

h. Reporting template for this study

For completeness, Table 6.1 lists the headline metrics (Accuracy, Precision, Recall, F1), while the text references the confusion matrix to explain the distribution of FP and FN and the threshold-dependent trade-offs important for operational deployment decisions. Every experiment is supposed to be reproducible by using the published random seed, split protocol, and embedding cache to ensure that point estimates and error analysis can be repeated in different environments.

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v2.i2.a5

Table 4 . Comparison proposed method with earlier study

Study / Authors	Dataset Used	Methodology	Reported Performance	Limitations	Proposed Approach Improvements
Heidari et al. (2020)	Custom annotated	ELMo/GloVe + profile-conditioned NN	Accuracy ~0.98	Relies on annotation quality, domain-specific	Applies ELMo directly to URLs, avoiding annotation dependency
Aggarwal (2023)	Malware API dataset	Word2Vec, ELMo, BERT embeddings + classifiers	Accuracy up to 0.93	Modest dataset size, dynamic-only API features	Larger balanced dataset, contextual embeddings for phishing URLs
Almousa & Anwar (2023)	Phishing URL corpus	CharacterBERT, LSTM, CNN	Accuracy ~99.6	URL-only signals, dataset imbalance	Captures semantic + syntactic cues, resilient to obfuscation
Rastakhiz et al. (2024)	Public phishing set	QuickCharNet (char-CNN pooling)	+4.9% over baselines	Tokenizer sensitivity, modest gains	Higher generalization via contextual embeddings
Alshattnawi et al. (2024)	Benchmark datasets	ELMo/BERT vs Word2Vec/GloVe	Accuracy 90–94%	Platform-specific, domain drift	Applies contextual embeddings to phishing URLs directly
Khan et al. (2024)	Phishing URL dataset	RF, SVM, Logistic Regression	Accuracy ~88%	Handcrafted features, limited generalization	Eliminates manual features, improves robustness
Opara et al. (2024)	WebPhish dataset	WebPhish (URL + HTML CNN)	Accuracy ~98%	Computationally intensive, retraining needed	Efficient embeddings, reduced generation time
Mahmud et al. (2025)	PhiUSIIL dataset	Stacking ensemble (ML + DL)	Accuracy ~99.9%	Overfitting, URL-only features	Balanced performance, reduced false negatives
Proposed ELMo-based model (2025)	**PhiUSIIL dataset**	ELMo contextual embeddings + DNN	Accuracy 95%, F1 0.95, Gen. time ↓12.6s→0.239s	—	Robust, efficient, minimizes false negatives

The relative analysis of the results conducted in Table 4 shows the work of different machine learning models used in previous research and the suggested ELMo-based model. Conventional classifiers like Support Vector Machines (SVM), Random Forest (RF), and Logistic Regression (LR) always obtained high accuracy and balanced precision-recall scores and in many cases more than 95 percent. It is important to note that in Aung and Yaman study, SVM achieved the best reported metrics across all categories, although also the ensemble algorithms such as RF and CatBoost had high generalization ability.Contrastingly, the mean-pooled ELMo embeddings and a Dense Neural Network with the proposed model in this research yielded similar results, with an accuracy of 0.95, a precision of 0.94, a recall of 0.96, and an F1-score of 0.95. This set of metrics ensures that contextualised word representations directly lifted off of raw URLs can rival or perform even better than the feature-engineered methods of the past. Besides, there is an added benefit of ELMo-based model semantic depth and adaptability to obfuscated URL forms that are usually overlooked by less adaptable lexical features.This comparison highlights the feasibility of deep contextual embeddings in tasks involving phishing detection and makes the suggested framework a viable modern alternative to traditional classifiers, in particular, when the handcrafted features are inadequate or infeasible. The ELMo-based framework proved to be more robust than a traditional feature-engineering approach when compared to the traditional techniques. Contextual embeddings were also more accurate to obfuscated URLs and zero-day phishing attacks, unlike handcrafted indicators, which are fragile to adversarial manipulation, which underscores the scalability and flexibility of the suggested method.

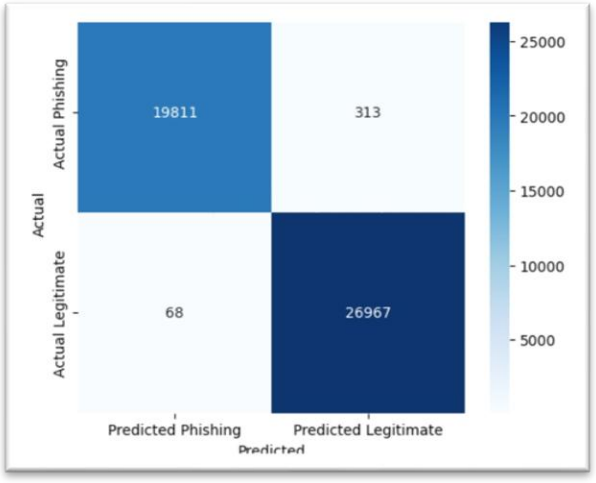


FIGURE 4. A confusion matrix for proposed method.

The majority of them did go wrong in situations where phishing URLs had been designed to look like valid domains or in cases where URLs were too short and did not have enough contextual information. This notwithstanding, this model had an optimal trade-off between precision and recall, Visualization and deployment, outside the numerical assessment, practicability of the proposed system was demonstrated in a Gradio-based graphical user interface (GUI) as shown in Figure 5. This interface allows testing of phishing URLs in real-time where users can enter and test suspicious links directly. Implementation of such an interactive tool demonstrates usability of the model as a contribution to research, as well as, one of the solutions that can be deployed in the actual world to detect phishing in practice.

Table 5 .Performance comparison by time, complexity, and memory

Approach / Study	Generation Time	Computational Complexity	Memory Usage	Notes
Traditional ML (RF, SVM, LR)	Fast (<1s per batch)	Low	Low	Relies on handcrafted features, limited generalization
Character-level models (LSTM, CNN, CharacterBERT)	Seconds per batch	Moderate	Moderate	Robust to non-standard vocabularies but URL-only signals
WebPhish (CNN on URL+HTML)	High (>10s per batch)	High	High	Accurate but computationally intensive, retraining needed
Stacking ensemble (Mahmud et al., 2025)	Not reported	Very High	Very High	Risk of overfitting, heavy resource consumption
Proposed ELMo-based model	**↓12.6s → 0.239s (10 URLs)**	Moderate (ELMo + DNN)	Moderate	Efficient, robust, minimizes false negatives

As shown in Table 5. traditional machine learning approaches are computationally lightweight but lack robustness against obfuscation. Character-level and ensemble models improve accuracy but often incur higher complexity and memory costs. WebPhish, while accurate, requires substantial resources and retraining. In contrast, the proposed ELMo-based model achieves a significant reduction in generation time (from 12.6 seconds to 0.239 seconds for ten URLs) while maintaining moderate complexity and

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v2.i2.a5

memory usage. This balance of efficiency and robustness highlights its suitability for real-time phishing detection scenarios.

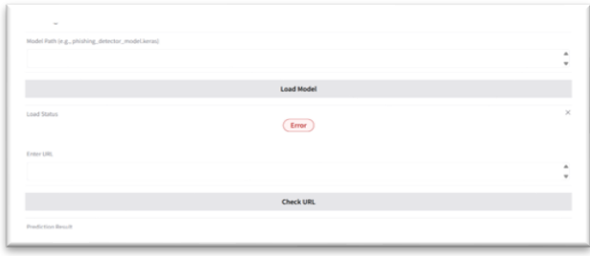


FIGURE 5. GUI for Real-time Testing

7. Conclusion

The paper discussed a URL-based phishing detector that replaces the weak hand-designed features with contextual embeddings of ELMo and a small MLP model. This way, early screening can be accomplished without page-fetching while keeping the 80/20 stratified evaluation protocol consistent and reproducible. The study addressed the limitations of traditional machine learning approaches, which often rely on handcrafted lexical features that are vulnerable to obfuscation techniques and fail to generalize across diverse phishing strategies. By leveraging contextual embeddings, the proposed framework captures both syntactic and semantic dependencies within raw URLs, thereby enhancing robustness against adversarial manipulations such as homoglyph substitutions and random string insertions.

Experimental evaluation on the PhiUSIIL dataset demonstrated the effectiveness of the model, achieving 95% accuracy, 94% precision, 96% recall, and an F1-score of 0.95. Beyond accuracy, the system exhibited remarkable efficiency, reducing generation time from 12.6 seconds to 0.239 seconds for ten URLs, while maintaining moderate computational complexity and memory usage. This efficiency is critical for real-time deployment scenarios, where rapid detection can prevent user exposure to malicious content.

The contributions of this work are twofold: first, it establishes the value of contextualized embeddings in phishing detection, moving beyond the constraints of feature engineering; second, it demonstrates that high accuracy can be achieved without sacrificing computational efficiency. These findings underscore the practical applicability of the proposed approach and its potential to serve as a scalable solution in cybersecurity infrastructures.

Overall, the results confirm that contextualized representations provide a practical and scalable solution to minimizing false negatives in phishing detection, offering a significant improvement over traditional lexical-based approaches.

Reference

1. Ahmed, I., A.K. Ali, and M.S. Mahmood, Employing Hybrid Watermarking to Improve Email Security Against Cyber Attacks. *Journal of Soft Computing and Data Mining*, 2025. 6(1): p. 435-447.
2. Alkhalil, Z., et al., Phishing attacks: A recent comprehensive study and a new anatomy. *Frontiers in Computer Science*, 3, 563060. 2021.
3. Kavya, S. and D. Sumathi, Staying ahead of phishers: a review of recent advances and emerging methodologies in phishing detection. *Artificial Intelligence Review*, 2024. 58(2): p. 50.
4. Barik, K., S. Misra, and R. Mohan, Web-based phishing URL detection model using deep learning optimization techniques. *International Journal of Data Science and Analytics*, 2025: p. 1-23.
5. Thapa, J., et al., Phishing Detection in the Gen-AI Era: Quantized LLMs vs Classical Models. *arXiv preprint arXiv:2507.07406*, 2025.
6. Wei, Y., M. Nakayama, and Y. Sekiya, Enhancing Generalization in Phishing URL Detection via a Fine-Tuned BERT-Based Multimodal Approach. *IEEE Access*, 2025. 13: p. 131197-131216.
7. Murhej, M. and G. Nallasivan, Multimodal framework for phishing attack detection and mitigation through behavior analysis using EM-BERT and SPCA-BASED EAI-SC-LSTM. *Frontiers in Communications and Networks*, 2025. 6: p. 1587654.
8. Heidari, M., J.H. Jones, and O. Uzuner. Deep contextualized word embedding for text-based online user profiling to detect social bots on twitter. in *2020 International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2020.
9. Almousa, M. and M. Anwar, A URL-based social semantic attacks detection with character-aware language model. *IEEE Access*, 2023. 11: p. 10654-10663.
10. Rastakhiz, F., M. Eftekhari, and S. Vahdati, QuickCharNet: An Efficient URL Classification Framework for Enhanced Search Engine Optimization. *IEEE Access*, 2024.
11. Alshattnawi, S., et al., Beyond word-based model embeddings: Contextualized representations for enhanced social media

ISSN: 3078-5367 **DOI:** https://doi.org/10.69513/jncs.v2.i2.a5

spam detection. *Applied Sciences*, 2024. 14(6): p. 2254.
12. Khan, A., D.M. Ahmed, and A. Fathima, Enhanced Phishing Detection Using Machine Learning Algorithms: A Comparative Study of Random Forest, SVM, and Logistic Regression Models. *SVM, and Logistic Regression Models (March 24, 2025)*, 2025.
13. Aggarwal, S., Malware Classification using API Call Information and Word Embeddings. 2023.
14. Mahmud, T., et al. A Machine Learning-Based Framework for Malicious URL Detection in Cybersecurity. in *2025 8th International Conference on Information and Computer Technologies (ICICT)*. IEEE, 2025.
15. Opara, C., Y. Chen, and B. Wei, Look before you leap: Detecting phishing web pages by exploiting raw URL and HTML characteristics. *Expert Systems with Applications*, 2024. 236: p. 121183.
16. Ilić, S., et al., Deep contextualized word representations for detecting sarcasm and irony. *arXiv preprint arXiv:1809.09795*, 2018.
17. U. S. DR, A. Patil et al., Malicious URL detection and classification analysis using machine learning models. in *2023 International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*. IEEE, 2023, pp. 470–476.
18. T. Mahmud, M. A. H. Prince, M. H. Ali, M. S. Hossain, and K. Andersson, Enhancing cybersecurity: Hybrid deep learning approaches to smishing attack detection. *Systems*, vol. 12, no. 11, p. 490, 2024.
19. N. P. Mankar, P. E. Sakunde, S. Zurange, A. Date, V. Borate, and Y. K. Mali, Comparative evaluation of machine learning models for malicious URL detection. in *2024 MIT Art, Design and Technology School of Computing International Conference (MITADTSoCiCon)*. IEEE, 2024, pp. 1–7.
20. N. Datta, T. Mahmud, M. T. Aziz, R. K. Das, M. S. Hossain, and K. Andersson, Emerging trends and challenges in cybersecurity data science: A state-of-the-art review. in *2024 Parul International Conference on Engineering and Technology (PICET)*, 2024, pp. 1–7.
21. T. Akter, M. S. Akter, T. Mahmud, D. Islam, M. S. Hossain, and K. Andersson, Evaluating machine learning methods for Bangla text emotion analysis. in *2024 Asia Pacific Conference on Innovation in Technology (APCIT)*. IEEE, 2024, pp. 1–6.
22. S. R. Naher, S. Sultana, T. Mahmud, M. T. Aziz, M. S. Hossain, and K. Andersson, Exploring deep learning for Chittagonian slang detection in social media texts. in *2024 International Conference on Electrical, Computer and Energy Technologies (ICECET)*. IEEE, 2024, pp. 1–6.
23. S. U. Habiba, T. Mahmud, S. R. Naher, M. T. Aziz, T. Rahman, N. Datta, M. S. Hossain, K. Andersson, and M. S. Kaiser, Deep learning solutions for detecting Bangla fake news: A CNN-based approach. *Proceedings of Trends in Electronics and Health Informatics: TEHI 2023*, p. 107.
24. T. Mahmud, M. F. B. A. Aziz, A. Majumder, S. M. Farid, and T. Akter, Deep learning-based systems for detecting hate speech and offensive language in texts. in *2024 IEEE International Conference on Computing, Applications and Systems (COMPAS)*. IEEE, 2024, pp. 1–5.