

Article

Real-Time Phishing Detection for Brand Protection Using Temporal Convolutional Network-Driven URL Sequence Modeling

Marie-Laure E. Alorvor * and Sajjad Dadkhah * 

Faculty of Computer Science, Canadian Institute for Cybersecurity, University of New Brunswick, Fredericton, NB E3B 5A3, Canada

* Correspondence: marie-laure.alorvor@unb.ca (M.-L.E.A.); sdadkhah@unb.ca (S.D.)

Abstract

Phishing, especially brand impersonation attacks, is a critical cybersecurity threat that harms user trust and organization security. This paper establishes a lightweight model for real-time detection that relies on URL-only sequences, addressing limitations for multimodal methods that leverage HTML, images, or metadata. This approach is based on a Temporal Convolutional Network with Attention (TCNWithAttention) that utilizes character-level URLs to capture both local and long-range dependencies, while providing interpretability with attention visualization and Shapley additive explanations (SHAP). The model was trained and tested on the balanced GramBeddings dataset (800,000 URLs) and validated on the PhiUSIIL dataset of real-world phishing URLs. The model achieved 97.54% accuracy on the GramBeddings dataset, and 81% recall on the PhiUSIIL dataset. The model demonstrated strong generalization, fast inference, and CPU-only deployability. It outperformed CNN, BiLSTM and BERT baselines. Explanations highlighted phishing indicators, such as deceptive subdomains, brand impersonation, and suspicious tokens. It also affirmed real patterns in the legitimate domains. To our knowledge, a Streamlit application to facilitate single and batch URL analysis and log feedback to maintain usability is the first phishing detection framework to integrate TCN, attention, and SHAP, bridging academic innovation with practical cybersecurity techniques.



Academic Editors: Tomotaka Kimura, Yu-Kuen Lai and Kouji Hirata

Received: 19 August 2025

Revised: 8 September 2025

Accepted: 16 September 2025

Published: 22 September 2025

Citation: Alorvor, M.-L.E.; Dadkhah, S. Real-Time Phishing Detection for Brand Protection Using Temporal Convolutional Network-Driven URL Sequence Modeling. *Electronics* **2025**, *14*, 3746. <https://doi.org/10.3390/electronics14183746>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: phishing detection; brand protection; Temporal Convolutional Networks (TCN); attention mechanism; Explainable AI (XAI); SHAP (Shapley Additive Explanations); URL sequence modeling; real-time cybersecurity; adversarial robustness

1. Introduction

Phishing attacks still represent one of the most prevalent and serious crimes in the cyber world, with adversaries increasingly using deceptive URLs to impersonate trusted brands and gain sensitive information from naive users. As phishing tactics mature, the need for detection approaches that are accurate and efficient, as well as explainable, must be at the forefront of development. Existing detection systems have implemented multi-modal methods such as HTML parsing, natural language processing (NLP), image-based logo identification, and detection techniques to identify impersonation attacks [1–3]. Although existing work has shown great promise with accurate detections, systems require usability improvements, especially with computational overhead, delays, and extensibility in low-capacity systems such as browser-based extensions or mobile applications [4].

Recent studies have demonstrated a decisive shift toward lightweight, URL-only detection models for phishing detection. Unlike multimodal models that consider webpage content, HTML parsing, natural language processing, and logo detection using pictures, the URL-only models consider the semantic and lexical structure of the text of the URL. This shift is also because of multimodal models, which achieve strong predictions, but suffer from substantial operational difficulties in practice. They incur large computational costs due to unnecessarily duplicated feature extraction pipelines [5], cannot scale in real-time environments such as browsers or email gateways [6], and have significant latency issues when competing with time from request-to-response, leading to ineffectiveness against zero-day phishing attacks [3]. Therefore, URL-only models benefit from faster inference times, better extensibility, more well-defined characteristics for low-capacity systems, and independence from language. In this study, we will expand our discussion of using Temporal Convolutional Networks (or TCNs) for sequence-based modeling. TCNs produce the best aspects of recurrent neural networks such as RNNs and LSTMs with the stability and parallelism of a convolutional network, which help stabilize gradient flow and with sufficient stacks of dilated and causal convolutions over multiple layers, they extend and permit the modeling of longer-range dependencies [7,8].

This project provides details of a full exploration of real-time phishing detection utilizing a character-level TCNWithAttention model. The model was trained on the GramBeddings dataset, which is a balanced dataset containing 800,000 phishing and legitimate URLs, and tested on the new, independent dataset—PhiUSIIL. The model includes an attention mechanism to focus on important character sequences used for legitimate URLs and phishing URLs, as well as using SHAP (Shapley additive explanations) for detailed, explainable output. The metrics for comparison with CNNs, BiLSTMs, and a fine-tune DistilBERT model included F1-score, precision, and inference time, with all tasks performing efficiently. Experimental results reveal that the TCNWithAttention model showed a better classification performance than CNN, BiLSTM, and BERT, with a test accuracy of 97.54% and sustained 86% precision for not-yet-seen phishing URLs.

Unlike CNN, BiLSTM, or Transformer baselines which emphasize raw accuracy, our TCNWithAttention model improves the field by linking stable long-range sequence modeling efficiency directly with transparent interpretability in an attention mechanism, and interpretable contributions using SHAP to enable both high accuracy and the potential for practical trust in real-time brand protection systems. To our knowledge, this is the first phishing detection framework to integrate TCNs with attention and SHAP for URL-only detection, combining efficiency, accuracy, and interpretability in a deployable real-time system.

From the study of this project, the following research questions were developed to frame its development and evaluation.

- To what extent can a TCN-based architecture accurately and efficiently learn sequential patterns from URL structures to detect phishing in a real-time situation?
- How does the TCNWithAttention model compare to the known CNN, BiLSTM, and BERT administrator frameworks, compared to the actual predictive performance, interpretability, and inference efficiency of each model?
- Can attention and SHAP values improve model transparency to support transformability and trust in real-time operational cybersecurity settings?

This project has made several important contributions to the field of phishing detection and brand protection. The key contributions of the project are outlined as follows:

- **Novel Architecture:** developed a lightweight TCNWithAttention model to detect phishing URLs with only URL sequences to address latency and scalability challenges of multimodal systems.

- **Comprehensive Benchmarking:** performed a comprehensive series of experiments comparing our model against CNN, BiLSTM, and BERT baselines demonstrating superior performance while operating in a near real-time fashion on CPU.
- **Explainability:** integrated attention visualizations and SHAP analysis to identify the URL component contributing to the model's predictions. This enhances trust and usability in a security analyst's workflow.
- **Deployable Solution:** developed a prototype deployable system using Streamlit that allows real-time classification, batch classification, bilingual support, and analyst feedback integration, bridging the gap between the academic models and industry deployable tools.

The overall structure of this project is organized in the following order. Section 2 presents a review of the related literature on phishing detection. Section 3 presents the motivation behind this project. Section 4 presents a research methodology and aims to recapitulate the framework within which the project was conducted. Section 5 presents the implementation of the model and system design which involves the training of models and demonstrates how to potentially implement it. Section 6 provides evaluation, testing and interpretability data. Section 7 presents the discussion on deployment and real-life system integration. Section 8 is the wrap-up of the project which presents the conclusion, key achievements, and future work.

2. Literature Review

2.1. Traditional Phishing Detection Techniques

Historically, phishing detection relied on blacklists and rules/signature-based approaches (e.g., Google Safe Browsing and Spamhaus) which were lightweight and easy to implement, but were inherently reactive methods which were brittle and easily circumvented by zero-day attacks and obfuscated phishing schemes [9]. Heuristic approaches (lexical rules, domain/WHOIS/hosting signals, and HTML signals) improved coverage but suffered from a lack of robustness to evasion (typosquatting, homograph substitutions, and scripted redirections) and required constant manual curation and updates [10,11].

Traditional machine learning (ML) methods (SVMs, decision trees, random forests, Naïve Bayes) increased generalizability, but required considerable feature engineering across URLs, HTML, and WHOIS metadata. Additionally, these systems often performed poorly on new distributions, especially in multilingual settings, or if the training data was stale or had a narrow focus [12,13]. These limitations were a primary factor influencing a transition to deep learning (DL)-based models, which accepted raw strings and learned the discriminative features directly from data [13].

2.2. Deep Learning for URL-Based Phishing Detection

Recent work has increasingly used deep learning (DL) to learn the lexical and styled patterns present in URL sequences through a DL model, reducing the number of manual features used and increasing resistance to adversarial construction (e.g., nested subdomain, obfuscated paths) [13]. As an example, CNNs (e.g., URLNet) have semi-supervised and learned locality at scale across multiple n-grams (token identifiers, brand spoofing tokens, and suspicious affixes and prefixes) in the character/word embedding, but limited at modeling time series and long-range dependencies [6]. Although RNNs/BiLSTMs provide good sequential context, they exhibit slow learning, impractical parallelism, and prolonged latency, rendering them less suitable for real-time deployments on CPU [14,15]. Conversely, the semantic modeling of Transformers/BERT is very strong, but the computational costs (e.g., GPU dominant processing and RAM and memory) of inference limit the case for deployability at the edge (e.g., browsers, gateways, and mobile) [16]. A compelling alterna-

tive is temporal convolutional networks (TCNs) which combine the speed and parallelism of CNNs with long-range temporal modeling through dilated causal convolutions and residual connections, transcending speed and stability to be more suitable in terms of character-level URL classification [5]. Recent studies outlined that TCN-based detectors performed better than RNN-based detectors with similar accuracy to BERT, but at several magnitudes better in terms of efficiency [7,8].

2.3. Multimodal and HTML-Based Phishing Detection

Multimodal systems (screenshots/logos/DOM/HTML) such as Phishpedia and KnowPhish have achieved very high levels of accuracy by utilizing rich visual and structural cues [1,2]. However, they suffer from rendering latencies, crawler intrusion, and infrastructure overheads, which limits their scalability and real-time deployment (edge plugins, mail gateways, and low-power devices). They are also affected by adversarial HTML obfuscation (JavaScript redirecting, DOM morphing, and CSS cloaking) [3,17]. Comparative studies have shown that although multimodal methods can be superior in offline/forensic assessments, URL-only systems are operationally better for inline protection because they are faster and simpler [13].

2.4. The Shift to URL-Only Detection

Recent studies have shown that URL-only models can equal and exceed the performance of multimodal methods in operating environments, mainly because they do not need HTML rendering and third-party lookups, allowing for sub-50 ms inference while parsing a page may take ≈ 800 ms [4,10]. URL-only systems are agnostic to language, resistant to crawler blocking, and also supportive of real-time explainability (token-level attributions); this favors deployment in browsers, SWGs, and MTAs. Further empirical evaluation (e.g., GramBeddings) has shown strong generalization with previously unseen URLs, supporting sequence-only modeling at scale [4,11].

2.5. Temporal Convolutional Networks (TCNs)

TCNs utilize stepwise dilated causal convolutions to gradually broaden the reception field exponentially while also preserving the temporal order. Furthermore, the residual connections stabilize gradients so that longer stacked convolutional layers can be prescriptively trained across training samples while enabling quick and parallel training and inference on CPU [5]. When it comes to phishing detection, TCNs are competitive or outperform RNNs while being competitive with BERT at a fraction of the cost/latency [7,8]. Additionally, TCNs can be integrated with attention mechanisms to accentuate contextual portions of URLs (e.g., deceptive subdomains and brand misuse). Using attention mechanisms' interpretable models may also assist in the triage process for analysts [18]. TCNs have complementary attributes for phishing detection in terms of accuracy, efficiency, and transparency which make it a strong candidate for brand protection and other types of live defenses.

2.6. Interpretability in Phishing Detection

There is also an increasing need for explainability in natural language processing models for operational trust, auditing, and regulatory compliance (i.e., explainable AI/XAI). Attention pooling is now commonly used as a signal and may provide more intuitive "where the model looked" signals along the URL. SHAP also provides additive model-agnostic attributions, which quantify "how much did each character/token contribute" to a model prediction [19]. Previous work had model explanations with either attention, LIME/SHAP, or as graph-based forms of explanations but with the tradeoff of either consistently relying on other knowledge bases, or they were a prolonging programming

implementation [2]. Attention and SHAP's direct explanation provide granularity and transference testing, with very small computational or human resource use given the URL-only pipeline, allowing analysts to check all informative cues in URLs like brand impersonation tokens, suspicious TLDs, and stuffing subdomains.

2.7. Research Gap and Contributions of the Research Project

Despite advancements, research gaps exist:

- *Latency and Deployability*: Existing multimodal solutions remain too resource intensive for real-time edge applications [1,2].
- *Lightweight and Interpretability*: The integration of URL-only models, that simultaneously provide speed, accuracy, and figurative explanation suitable for SOC workflows is practically non-existent [5,13].
- *TCNs not Fully Utilized*: Current evidence supports TCNs for lengthier URLs and patterns, but they are underused compared to LSTMs/BERT [8,10].
- *Comparative Benchmarking*: Majority of the studies are not truly comparative—same datasets/tokenization but frequently omit inference latency and model size and always omitted for deployment [11,14].
- *Zero-Day Robustness*: More work is needed related to adversarial/obfuscated URL variants (e.g., homoglyphs, brand-mixing subdomains, dubious TLD) [7].

This work addresses the research gaps by proposing a URL-only, lightweight TCN-WithAttention framework; controlling performance benchmarks rigorously against CNN, BiLSTM, and BERT; proposing attention and SHAP for character-based interpretability; and demonstrates a CPU only real-time edge deployment for commercial brand protection hood.

2.8. Comparative Analysis of Related Work

Existing phishing detection models fall into one of two types: multimodal (e.g., Phishpedia and KnowPhish) and URL-only systems. Multimodal systems appear to best utilize HTML, logo images, and screenshots as part of their phishing detection process, often achieving benchmarks at higher accuracies but requiring higher latency, and they have crawler dependencies and resource overheads leading to challenges in real-time scalability. URL-only systems, which utilize CNNs, BiLSTMs, TCNs, or other Transformer variants, reach competitive accuracy with reduced latency and are therefore more suited for use in browsers, secure email gateways, and mobile applications. Tables 1 and 2 summarize the comparisons presented by showing the different model features, datasets, performance, and trade-offs.

Table 1. Comparative analysis of related work.

S/N	Author	Proposed Model	Model Features	Dataset	Dataset Availability	Features Extraction	Performance	Strengths	Limitations
1	Opara, C., Wei, B., and Chen, Y. [3]	HTMLPhish: Enabling Phishing Web Page Detection by Applying Deep Learning Techniques on HTML Analysis	Character- and word-level HTML token embedding, CNN + Dense layers	47k legitimate and 4.7k phishing HTML pages from Alexa and PhishTank	Unavailable	Raw HTML (character-level + word-level)	Accuracy: 98.1%	Uses the raw HTML and does not require feature engineering.	Sensitive to HTML noise and does not generalize to image-based phishing
2	Yun Lin, Ruofan Liu, Dinil Mon Divakaran, Jun Yang Ng, Qing Zhou Chan, Yiwen Lu, Yuxuan Si, Fan Zhang, and Jin Song Dong [1].	Phishpedia: A Hybrid Deep Learning-Based Approach to Visually Identify Phishing Webpages (2021)	Logo matching, URL, visual screenshot analysis. Faster R-CNN and Siamese Neural Network.	PhishTank, OpenPhish and Alexa	Available online: https://sites.google.com/view/phishpedia-site/home (accessed on 7 May 2025)	Domain, logo, and visual screenshot	Accuracy: 99.2%. Precision: 98.2% Recall: 87.1%	Robust to obfuscation, and logo-based explainability.	False positives due to logo collision, and good quality logo set required.
3	Yuexin Li, Chengyu Huang, Shumin Deng, Mei Lin Lock, Tri Cao, Nay Oo, Hoon Wei Lim, and Bryan Hooi [2]	KnowPhish: Large Language Models Meet Multimodal Knowledge Graphs for Enhancing Reference-Based Phishing Detection (2024)	LLM + Multimodal Knowledge Graph, HTML, logos, domains. Text-based CRP Classification. Brand Extractor. Domain Check.	KnowPhish Knowledge Base (Wikidata + Tranco + Whois) TR-OP SG-SCAN	https://github.com/imethanlee/KnowPhish (accessed on 7 May 2025)	HTML— JavaScript and CSS blocks. Credential-taking elements, logo, text brand, and domain	Accuracy: 92.49% Precision: 97.84% Recall: 86.90% F1-score: 92.05%	Works well with obfuscated pages; scalable across 20k+ brands.	Requires third party data for parallel URLs; false positives as a result of brand overlaps.

Table 1. Cont.

S/N	Author	Proposed Model	Model Features	Dataset	Dataset Availability	Features Extraction	Performance	Strengths	Limitations
4	Ahmet Selman Bozkir, Firat Coskun Dalgic, Murat Aydos [4]	GramBeddings: A New Neural Network for URL-Based Identification of Phishing Web Pages Through N-gram Embeddings	N-gram embedding of URLs, 1D CNN + BiLSTM + Attention	GramBeddings-DS: 400k phishing URLs from Phish-Tank/OpenPhish, 400k legit via crawler	Unavailable	N-gram (1,4,5,6) on URL tokens	Accuracy: 97.85%, F1: 97.93%	Lightweight, language agnostic, and works well on newly seen URLs.	Frequent retraining required, and sensitive to adversarially modified URLs.
5	Lixia Xie, Hao Zhang, Hongyu Yang, Ze Hu, Xiang Cheng [7]	A Scalable Phishing Website Detection Model Based on Dual-Branch TCN and Mask Attention	URL and HTML features. Parallel TCN branches. Mask Attention for contextual enhancement.	Crawled from PhishTank and Alexa	Available online: https://github.com/su-per-go/crawling_url (accessed on 12 May 2025) https://www.kaggle.com/datasets/haozhang1579/crawling2024 https://github.com/su-per-go/feature_extra https://phishtank.org/ https://www.alexa.com/topsites (accessed on 15 May 2025)	Parallel contextual HTML/URL features	Accuracy: 98.7% F1-score: 98.5%	Scalable to large-scale data and effective in capturing global and local dependencies	Sensitive to noisy HTML and computationally expensive
6	Nguyet Quang Do, Ali Selamat, Ondrej Krejcar, Hamido Fujita [8]	Detection of Malicious URLs using TCN and Multi-Head Self-Attention Mechanism	Sequential URL-based features. Combines TCN with Transformer-like self-attention.	Ebbu2017 Phishing Dataset (GitHub)	Available online: https://github.com/ebubekirbbr/pdd/tree/master/input (accessed on 15 May 2025)	URL structure + position encoding	Accuracy: 97.6% Precision: 96.9%	Strong performance on limited URL data, lightweight and effective.	Limited feature diversity (only URL) and less effective with multimodal features.

Table 1. Cont.

S/N	Author	Proposed Model	Model Features	Dataset	Dataset Availability	Features Extraction	Performance	Strengths	Limitations
7	Ali Aljofey, Saifullahi Aminu Bello, Jian Lu, Chen Xu [18]	Comprehensive phishing detection: A multi-channel approach with variants TCN fusion leveraging URL and HTML features	Handcrafted URL and HTML features. Parallel TCN branches with dense fusion	PhishStats and Alexa Top 1M	Available online: https://www.alexa.com/topsites https://phishstats.info/ (accessed on 19 May 2025)	Manual URL + HTML features	Accuracy: 98.21% Precision: 98.32% Recall: 97.93%	Balanced feature modeling and strong fusion-based architecture	Static handcrafted features and limited adaptability to zero-day threats.

Table 2. Comparative summary of research papers in 2025.

S/N	Author	Proposal	Approach	Used URL	Model Used
1	Lixia Xie, Hao Zhang, Hongyu Yang, Ze Hu, Xiang [7]	Dual-Branch TCN + Mask	Uses raw URLs and HTML features for phishing classification using a dual TCN branch, one for each input modality.	URL	TCN
2	Ngoc Quang Do, A. Selamat, O. Krejcar, H. [8]	TCN + Multi-Head	URL-only phishing detector using TCN + Multi-Head Self-Attention, emphasizing temporal dependencies	URL	TCN
3	Ali Aljofey, Saifullahi Aminu Bello, Jian Lu, Chen Xu [17]	Multi-Channel TCN Fusion	Both URL and HTML hand-crafted features fused using multiple TCN branches.	URL	TCN
4	Sneha Baskota [14]	Phishing URL Detection using Bi-LSTM	Focuses on URL sequences via Bi-LSTM, confirming continued emphasis on URL-based modeling.	URL	Bi-LSTM
5	Wenye Guo, Qun Wang, Hao Yue, Haijian Sun, Rose Qingyang Hu [10]	Graph-Based ML with Loopy Belief Propagation	Integrates URL structure with network-level features—again URL-centric.	URL	Loopy Belief Propagation
6	Muhammad Fahad Zia; Sri Harish Kalidass [12]	Campaign Detection via Unsupervised ML (Web Phishing Net)	Detects phishing campaigns through URL analysis.	URL	Unsupervised ML
7	Rina Mishra, Gaurav Varshney [11]	Brand Domain Identification	Focuses on domain-based features for brand-related phishing detection—URL-rooted but domain-centric.	URL	Domain Heuristics
8	Jinyang Zhou, Kun Zhang, Anas Bilal, Yu Zhou, Yukang Fan, Wenting Pan, Xin Xie and Qi Peng [13]	CSPPC-BiLSTM	N-gram embedding approach using only URLs, with CNN-BiLSTM-Attention layers.	URL	CNN-BiLSTM-Attention
9	Cao, T., Huang, C., Li, Y., Huilin, W., He, A., Oo, N., and Hooi, B. [18]	PhishAgent	Uses URL, visual, and DOM features in a multimodal ensemble model.	Patial URL	Multimodal Ensemble (Visual, DOM, URL)
10	Yu-Hung Chen; Jiann-Liang Chen; Shih-Ping Chiu [16]	AI-Enhanced GAN for Robust Phishing Detection	Embeds URLs using Transformer-based encoders in a GAN framework for robust classification.	URL	Transformer + GAN

2.9. Comparative Summary of Research Papers in 2025

It is well established by current studies that a strong trend to rely more on URL-only deep learning models is emerging, especially for systems utilizing TCNs as the model of choice for efficiency, long-range sequences, and stable training. TCNs will appear to be the top performing models by consistently balancing accuracy, speed, and interpretability beyond RNNs with matching Transformer accuracy at less intensive resource levels. The tables go on to indicate that although multimodal models are still useful for forensic or offline analyses, we can expect a forward trend towards URL-only models, particularly with prototypical TCN-based systems for real-time brand protection and zero-day phishing detection systems.

As outlined above, we have converged in 2025 on URL-only Deep Learning Models for phishing detection, with TCNs being the most favorable amongst these, given their long-range dependency capability, as well as performance in terms of accuracy, speed, and scalability. The multimodal approaches such as PhishAgent, that use DOM structures or rendered screenshots, suffer from limited scalability, latencies with larger sample sizes or in live environments. In contrast, TCN-based models and, to a lesser degree, BiLSTM-based models, always achieve competitive accuracy while using less overhead. With attention mechanisms, Transformer-type variations, and GAN-type approaches are still being introduced, many of the proposals are not operationally feasible due to being computationally expensive, or otherwise impractical. Overall, whilst we can see the trajectory of the literature having strongly shifted towards URL-only detections, in terms of being lightweight, interpretable, and scalable, this supports the overall trajectory of this project in implementing a TCNWithAttention architecture.

3. Motivation

Phishing is still one of the most multiply present and most damaging types of cyber-crime, and cybercriminals show increasing interest in impersonating brands using sites that are designed to be visually and structurally similar to legitimate domains. Phishing detection has typically relied, in various combinations, on features such as page meta characteristics, HTML structure, the analysis of logo images, and other methods of full content inspection. These methods can work well, especially in offline contexts, but they often computationally intensive and cannot be used for real-time identification of phishing sites in tools such as browser-based detection plug-ins, or in lightweight API services. In addition, modern phishing techniques have begun to use a range of cloaking techniques and may deliberately obfuscate the meta characteristics of a site. This has the potential to greatly hinder the capability of these traditional approaches [1,2].

Recently, the literature has started to emphasize and encourage an alternative approach to phishing detection based on URL-only detection approaches, in which models can consider the structure and token patterns of URLs without downloading web page contents: for example, Do et al. (2025) [8] and S. Baskota (2025) [14] reported that sequential neural models were capable of learning useful representations from the character-level encodings of URLs, finding deceptive patterns indicative of phishing attacks in the process. Other studies, Bozkir et al. (2023) [4] and W. Guo et al. (2025) [10], further reported that URL-based models could achieve efficient real-time processing speeds and yet maintain reasonable levels of detection accuracy.

The primary value proposition to this project is to capture a version of accuracy, inference speed, and efficient use of computing resources, while ensuring some level of interpretability and transparency in the prediction. This is important for real-time applications, specifically for prevention using browser-based integrations or API-assumed

microservices, as the downloading of full web content is fundamentally impractical and any significant image processing is infeasible.

While CNNs, BiLSTMs, and Transformers (i.e., BERT) are state-of-the-art deep learning models for classification, they could not be deployed in real-time due to their enormous parameter sizes, latency, and lack of interpretability [13,16,18]. TCNs, on the other hand, are a viable option. TCNs are able to learn temporal dependencies in the URL sequences, operate on lower latency than recurrent networks, and have a smaller computational footprint than Transformer-based approaches. Moreover, TCNs with attention are able to extract and visualize parts of the URL it relies on for its prediction helping to build trust and trustworthiness while explaining the model's predictions in the context of Black and White Box Explainability for Cybersecurity.

Addressing the need for real-time phishing detection tools in brand protection that requires TCNs with attention could serve as a better alternative than more traditional architectures in terms of efficiency, accuracy, generalizability, and explainability. The aim was to develop a URL-only phishing detection framework which is lightweight and fast, as well as providing explanations on how and why the URL was classified as phishing.

4. Research Methodology

This study implemented a structured and iterative methodology that progressed through exploration to implementation. This project first examined multimodal detection of logos using image recognition and natural language processing (NLP), but due to scalability and latency challenges posed in the current literature, the project focused on a lightweight URL-only detection system. The refined methodology examined character-level sequence modeling of URLs, which commensurate with recent advances in phishing detection emphasizing lightweight, deployable models.

A. Phase 1: Initial Research and Scope Definition

This paper initially explored multimodal brand protection modalities encompassing webpage logo, content, and metadata. The literature review and preliminary results confirmed that URL-based phishing detection have proven to exceed scalability, accuracy, and real-time application of other methods. This produced a primary research question: could lightweight sequence models provide scalable practical and interpretable phishing detection?

B. Phase 2: Dataset Selection and Preprocessing

There were two publicly available datasets used in the study:

- *GramBeddings Dataset* [4]: a balanced dataset of 400,000 phishing and 400,000 non-phishing URLs, for character-level sequence modeling. The GramBeddings dataset was the dataset used for training and internal validation.
- *PhiUSIIL Dataset* [20]: a benchmark dataset composed of real-world phishing and benign URLs, used solely to externally validate and test robustness of models.

As a first step in data analysis, select datasets underwent preprocessing including URL cleaning, deduplication, and tokenization. Each URL was encoded as a 128 character fixed-length sequence using a custom character-level tokenizer, to ensure uniform representation of input data.

C. Phase 3: Model Development and Comparison

A total of four deep learning models were built for comparison purposes:

- CNN: A 1D convolutional neural network to serve as a traditional baseline.
- BiLSTM: A bidirectional long short-term memory network will be used to model the sequential nature of URL data.

- BERT: A Transformer-based language model will be used to fine-tune contextual features of raw URL text that will be extracted.
- TCN with Attention: A residual, dilated convolutional network, which was augmented with an attention pooling layer, that improved interpretability and model performance.

All models were trained on GramBeddings and evaluated based on inference speed, scalability, accuracy, precision, recall, and F1-score.

D. Phase 4: Evaluation and Comparative Analysis

This focused on a comparison of the four models: Temporal Convolutional Networks (TCN) with Attention, Convolutional Neural Networks (CNN), Bidirectional LSTM (BiLSTM), and BERT. Each model was compared by means of internal validation (i.e., the GramBeddings test split) and external evaluation (phiUSIIL dataset) on the comparative metrics, including confusion matrices, ROC curves with AUC, and classification heat maps, to gain the evidence of in-sample accuracy and out-of-sample generalization.

E. Phase 5: Interpretability and Explainability

To increase transparency, user trust, and model interpretability, the Temporal Convolutional Network (TCN) model included attention pooling and SHAP (Shapley additive explanations). The attention pooling mechanism displayed character-level highlights in a URL that the model identified as significant to making its prediction. SHAP also explained that values will be calculated to understand the quantitative feature contributions towards the model output. The combination allows both researchers and end users to visualize why a URL is classified as phishing or legitimate. The interpretability layer informed model auditing, amplified user trust in automated decisions, and encouraged insights for improvement as the model matures over time.

F. Phase 6: Real-Time System Integration

A prototype was implemented in Streamlit to classify one URL, or batch classifying many URLs in real time. The system included features such as bilingual support (English and French), logging both the predictions and any relevant feedback to an SQLite database, and the ability to log and visualize the SHAP values in our prototype system. The system was specifically designed and optimized to be deployed in CPU-only settings, allowing real-time deployment in low-resource or enterprise environments.

G. Phase 7: Future Work

The methodology also has the potential to be expanded in the future with features such as multilingual support, training with internationalized domain names (IDNs), adversarial robustness, and potentially deploying the model in a web browser. Although these functionalities were not completed in this research study, they make part of our plan for future work.

5. Design and Implementation

5.1. Problem Analysis

Phishing is one of the most widespread and damaging types of cyber-crime. Attackers often use various types of deception in URLs to trick users into providing credentials or downloading malware. Conventional blacklist-based approaches do not work against zero-day phishing attacks, and even model-based approaches that use HTML, images, or network analysis have latency and scalability issues. This work presents a lightweight, character-level phishing URL model, built with Temporal Convolutional Networks with Attention. The model was trained in leveraging the GramBeddings and PhiUSIIL datasets and is intended to be scalable, interpretable, and deployed in real-time environments.

5.2. Dataset Preparation

5.2.1. GramBeddings URL Dataset

The GramBeddings URL Dataset was selected for model training and evaluation, due to it being a balanced and diverse dataset. The dataset contains 800,000 URLs, split evenly into 400,000 phishing and 400,000 benign URLs [4]. The phishing URLs were collected from Phishtank and OpenPhish and span from May 2019 to June 2021, which encompasses both long-term and periodic sampling, and determination for similarity reduction, to develop better representativeness of samples. The benign URLs were layered and crawled to eliminate noise from the semantic class of interest. URLs were initially sampled from Alexa, and then iteratively crawled from the 20 most popular websites in 20 countries and randomly chosen [4]. The study's position on the sampling strategy adhered to sampling to achieve semantic diversity in URLs and developed adequate generalizability from industry and geography.

The dataset was preprocessed by removing duplicates and reducing domain-level overlap in a manner that increases the potential to generalize. The URLs in the dataset were selected specifically for usage in character-level modeling tasks and were released as part of the GramBeddings project [3,4]. The data from the GramBeddings [1] dataset was used for model training and evaluation. The raw files of data train.csv and test.csv contain labeled samples of URLs indicating if they are "Phishing" or "Legitimate". For consistency, all labels were re-stated using binary values (1 indicates phishing, and 0 indicates legitimate). The dataset was pre-processed by removing erroneous entries from the data and applying a custom clean script that ensured consistent quoting around URLs. A character-level vocabulary was derived from the training dataset to encode each URL into a fixed length (128 character) numeric sequence, reserving 0 for padding. The final dataset consisted of a train set of 639,999 samples, test set of 159,999 samples, total of 799,998 samples and vocabulary size of 70 characters including padding.

The dataset also achieves distributional equivalence at several levels of analysis: URL lengths for both phishing and benign classes average about the same, which condones potential bias from structural differences. The ratios of generic TLDs (gTLD) and country-code TLD (ccTLD) are similar among the phishing and benign classes, with phishing including about 60.10% .com domains and ccTLDs 11.82%, while the benign set contained 52.17% .com domains and 12.04% ccTLDs (X).

Linguistic bias also exists; as different languages are represented, rates are likely skewed against steep frequency of English-language domains. However, this dataset benefits from stratifying sources from 20 countries that have varying domain categories.

5.2.2. PhiUSIIL Phishing URL Dataset

The PhiUSIIL Phishing URL Dataset is an independent phishing dataset from the UCI Machine Learning Repository [2] that contained labeled phishing and legitimate URLs [2]. The dataset was used for out-of-sample use only and was used exclusively for external evaluation on an unseen real-world benchmark. It contained 235,795 instances, comprising 134,850 legitimate and 100,945 phishing URLs. It manually curated benign and phishing URLs from multiple sources that can be used for real-world validation. In addition, this independent external dataset was used to evaluate the generalization capability of the model that was previously trained. For consistency with GramBeddings formats, the label was reversed to the same order as GramBeddings. PhiUSIIL's original reverse argument (0 = phishing, 1 = benign) was converted to (1 = phishing, 0 = legitimate). The use of PhiUSIIL as an independent benchmark tool evaluated robustness, particularly against zero-day phishing URLs, and also obfuscated structures that are not present in the training data.

5.3. Data Encoding and Tokenizing

A custom tokenizer was built to convert characters to numeric indexes. Character indices were used to encode URLs and pad them to a set length of 128. The tokenized tensors were stored in .pt so that the tensors can be loaded immediately, at runtime, during training. The vocabulary, char2idx, was saved to a .pkl file for consistency across training, evaluating, and inference. Preprocessed tensors, train_X.pt, train_y.pt, test_X.pt, and test_y.pt were created for quick model input. A fixed 128 length was established for both maximum sequence lengths for consistency, performance, and computational efficiencies; overall, character length statistical descriptors for the training and evaluation datasets indicated that 95% of URLs in GramBeddings were less than 128 characters, whereas 93% in PhiUSIIL were shorter than 128 characters, which means that truncation only affected a few of the exceptionally long URLs, which are the outliers. Often, very long URLs represent either long, automatic query-generated entities, or more questionable brand-derived domains. All shorter URLs moved to length 128 were padded with a reserved token (index 0), which gives the entirety of the input structure a consistent matrix. Padding is not detrimental to model learning, because the padding token embeddings are assigned to the value of zero and masked during convolution; therefore, the network will only consider the indexed character positions that matter. For adaptive dilation strategies, the TCN architecture has fixed dilations at each layer, allowing the receptive field of the model itself to be expanded exponentially. This allows us to refer, again, to URLs as using information from either short range or long-range features; for example, the tokens “-login” will occupy the similarly short-range embedding representations as other length features, such as the link in “secure-amazon” which would possibly be followed by layers of malicious features. Although variable-length or adaptive padding were not implemented in this model, whenever URLs are approaching maximum length, the current dilation architecture ensures that the risk of losing critical features is effectively mitigated. As part of future work, this can be enhanced by implementing dynamic sequence models that adaptively adjust sizes or segment longer URLs, while still preserving semantic integrity.

5.4. Model Architecture

The proposed phishing detection system utilizes a Temporal Convolutional Network (TCN) architecture, with an attention mechanism to accept character-level URLs. The architecture of the model and flow of data through its components are shown in Figure 1.

- *Input:* The model accepts tokenized URLs as a sequence of integers which represent the characters in the URL. All the inputs are the same length (padded if necessary).
- *Embedding layer:* The embedding layer takes character indices and turns them into dense vectors of a fixed size of 64. This layer provides syntactic patterns and positions along a URL.
- *Residual TCN Blocks:* This block is where the power of Temporal Convolutional Network (TCNs) happens. They allow the model to learn long-range sequential dependencies in a causal and hierarchical way. Each of the residual blocks is organized in such a way that it has two 1D convolutional layers (with dilated convolution), which allows the kernel to cover a greater range of inputs without needing to increase kernel size and padding ($\text{padding} = (\text{kernel_size} - 1) \times \text{dilation}/2$). In addition, it helps to make sure that the length of the output sequence is the same as the input. Therefore, the input can remain the same when passed through the combination of two convolutional layers. In addition, there will be batch normalization, which stabilizes training because it normalizes the activation along the batch dimension. Finally, there is ReLU activation and dropout for regularization (by preventing overfitting to the training data by randomly zeroing out activations, so the model does not make

strong dependencies on certain activations during training iterations). The second convolutional layer is justified as follows. If the numbers of input channels $\neq$ output channels, a 1×1 convolution will change the dimensions for residual addition; otherwise, an identity mapping will suffice. Residual connections help the gradient to flow through the network and reduce the impact of vanishing gradients, making training deeper architectures effective. This captures multi-resolution temporal patterns from a sequence of chunked character-encoded URLs. Each of the dilations ensures that there is an enlargement of the receptive field to capture long-range dependencies in the sequential input.

- *Attention Pooling:* This module adds interpretability and focus. Instead of just using the final hidden state as in RNNs, it weights all the time steps in the sequence using learned importance scores. After extracting the temporal features with multiple TCN layers, attention pooling is performed across the temporal dimension. This module assigns attention weights to each time step by learning a weight (attention score) for each time step, which together produces a context vector as a weighted sum of the sequence features. This will allow the model to focus on the informative regions of the URL (with respect to classification), such as unusual subdomains or brand impersonation. The module uses an attention mechanism to weigh important characters or regions of the URL based on classification contexts. The attention weights are visualized to explain which characters the model attributed to using SHAP and attention plots.
- *Fully Connected Layers:* Following the Embedding Layer, which converts each character index to a vector of embed_dim size and correctly applies padding tokens with padding_idx = 0, the TCN stack builds multiple TCN layers with increasing dilation (expanding receptive field), so that each block can look at a wider temporal context. The Attention Layer is then applied to the output of the last TCN block. The output is a context vector that collapses the attended features into a scalar score (logit), which is passed as input to a fully connected feedforward network with LayerNorm, which is normalization across features, ReLU activation with dropout as a regularized classifier, and finally a linear layer that outputs only a single value. A sigmoid activation is used for training purposes for binary classification of phishing vs. legitimate.

5.5. Model Training

The TCN model with optimizations was implemented with PyTorch, and trained using only CPU hardware, demonstrating that it is deployable on lower resource constraints. The relevant configurations are presented in Table 3. The model was trained with the GramBeddings dataset, along with the following hyperparameters.

Table 3. TCN training configuration.

S/N	Parameter	Value
1	Optimizer	Adam with weight decay of 1×10^{-5}
2	Learning Rate	0.0005
3	Batch Size	128
4	Epochs	8
5	Loss Function	Binary Cross Entropy with Logits (BCEWithLogitsLoss)
6	Hardware	CPU
7	Framework	PyTorch
8	Embedding Dim	64
9	Max URL Length	128 characters

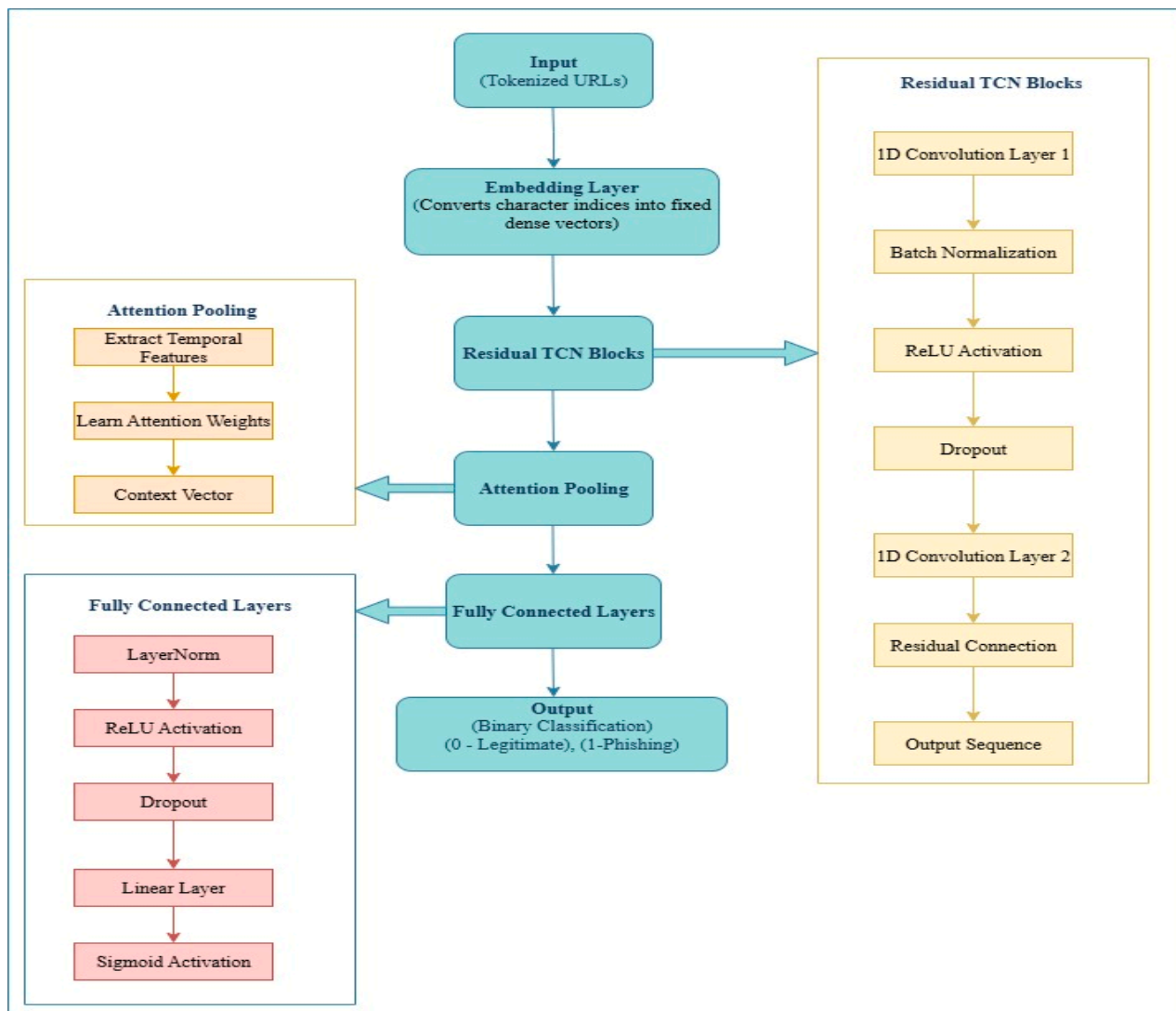


Figure 1. Architecture of proposed TCN model.

Training was conducted with the GramBeddings dataset. In every epoch, the model navigated the binary classification loss using Adam. Evaluation of the test set was completed at the end of every epoch to track performance metrics. The training was conducted on a CPU-based machine using PyTorch. Model checkpoints were saved whenever the F1-score of the validation set was improved upon. Performance metrics calculated while training consisted of Accuracy, Precision, Recall, F1-score. The final model with the highest F1-score was named `tcn_best_model.pt`.

5.6. Algorithm 1: Phishing Detection with TCNWithAttention

The Algorithm 1 further elaborates step by step analysis of the phishing detection using TCNWithAttention

Input:

- URL dataset $D = \{(url_1, label_1), (url_2, label_2), \dots, (url_n, label_n)\}$
- Pretrained character-to-index dictionary `char2idx`
- Max sequence length `MAX_LEN`

Output:

- Binary prediction $\hat{y} \in \{0 \text{ (Legitimate)}, 1 \text{ (Phishing)}\}$
- Explanation via SHAP + Attention Weights

Algorithm 1: TCNWithAttention Phishing Detection (D, char2idx, MAX_LEN)

```

1: Initialize model  $M \leftarrow \text{TCNWithAttention}()$ 
2: Load tokenizer  $T \leftarrow \text{CharTokenizer}(\text{char2idx}, \text{MAX\_LEN})$ 
3: Initialize optimizer, loss function (Binary Cross Entropy)
4: For each epoch do
5:   For each (url_i, label_i) in dataset D do
6:     sanitized_url  $\leftarrow \text{Sanitize}(\text{url}_i)$ 
7:     tokenized_seq  $\leftarrow T.\text{encode}(\text{sanitized\_url})$  # Pad/truncate
8:      $X_i \leftarrow \text{Embed}(\text{tokenized\_seq})$  # Character-level
embeddings
9:      $H_i \leftarrow M.\text{TCN}(X_i)$  # Residual dilated TCN
blocks
10:     $c_i \leftarrow M.\text{Attention}(H_i)$  # Attention pooling
11:     $\hat{y}_i \leftarrow \text{Sigmoid}(\text{FC}(c_i))$  # Output prediction
12:    loss  $\leftarrow \text{BCE}(\hat{y}_i, \text{label}_i)$ 
13:    Backpropagate loss and update weights
14:   End for
15: End for

```

5.7. CNN, BiLSTM, and BERT Baselines

To assess the performance of the proposed phishing detection system, we implemented three additional models to serve as baselines: CNN, BiLSTM, and DistilBERT. Each of these models was trained and evaluated on the same input representation and dataset as GramBeddings and PhiUSIIL, ensuring as fair as possible training, in addition to treating hyperparameters and explainable parameters the same.

5.7.1. Convolutional Neural Network (CNN)

The CNN model uses URLs as sequences of characters, allowing the model to maintain morphological and structural features of phishing links. The model's architecture is specified in `cnn_model.py` and includes the following key layers:

- *Embedding Layer:* Maps the input character indices to a 64-dimensional dense vector space. The total number of characters to assemble the vocabulary included printable ASCII characters and special tokens, resulting in 70 total characters. This reduces the input from shape [batch, seq_len] to [batch, seq_len, embed_dim].
- *1D Convolutional Blocks:* Multiple 1D convolutional layers are stacked to allow the model to extract local n-gram snapshots, such as the first Conv1D layer containing 128 filters with kernel size three, or trigrams, which allow for some number of words to be captured sequentially. Subsequent levels of convolutional layers further extract even more abstract features of the character sequences. Each convolutional layer is immediately followed by a ReLU activation and a dropout (0.1) layer to further allow for generalization.
- *Max Pooling Layer:* This layer is applied over the temporal dimension to reduce the overall dimension of the sequence while maintaining only the most important aspects.
- *Flatten Layer:* This layer flattens the 3D tensor to a 2D tensor [batch, features] that can subsequently be used for the fully connected layer.
- *Fully Connected Layers:* A dense layer that collapses the feature space down to 128 is included, followed by a ReLU activation, dropout and a final sigmoid for the classification layer to provide a binary classification output.

5.7.2. Bidirectional Long Short-Term Memory (BiLSTM)

The BiLSTM model seen in `bilstm_model.py` used both the past and future teaching context of the URL sequences. The ways it was structured were as follows:

- *Embedding Layer*: Like the CNN, the embedding layer changed character indices into 64 dimension vectors.
- *Bidirectional LSTM Layer*: This was a single BiLSTM layer with 128 hidden units for each direction, allowing the model to learn contextual dependencies for the URL sequence from both left-to-right and right-to-left. Employing both directions was useful for our training sets, since we want to capture suspicious patterns of interest, such as mirrored domains or substrings like ‘bad’ within longer substrings located anywhere in the URL.
- *Global max-pooling and attention mechanism*: We then pooled the LSTM outputs to extract a fixed-length (the original URL length) representation of the URL. Also note that we had a customized attention mechanism added to our model to learn how to aggregate a set of time steps with the weights we had learned.
- *Fully Connected Layers*: The pooled or attention-weighted vector was then passed along fully connected layers with 128 units with ReLU, dropout, then sigmoid as an output layer with binary classification.

5.7.3. DistilBERT Model

A Transformer-based model with DistilBERT was created, which functioned as a lightweight version of BERT, to assess the utility of using pre-trained language models to classify phishing URLs. The architecture and training logic are defined in the `bert_model.py` and `bert_train.py` scripts. This network type requires a GPU and increases training time considerably.

- *Tokenizer*: The URLs are passed into the model as raw strings. We tokenized these raw strings using the HuggingFace DistilBERT tokenizer which uses a subword tokenization scheme with WordPiece vocabulary.
- *DistilBERT Encoder*: The pre-trained DistilBERT model takes tokenized URLs as input and outputs contextualized embeddings, e.g., `[l, s_len, hidden_dim]`, where `hidden_dim = 768`. For our purposes, we are extracting the contextualized embedding for the [CLS] token from the last layer of the encoder, which is usually for classification.
- *Dropout and Classification head*: A dropout layer of 0.3 drop rate was used to regularize the output and added to a fully connected layer with one neuron (sigmoid activation), producing a binary classifier.

5.8. Novelty of the Approach

While CNN, BiLSTM, and Transformer-based methods have been widely used in phishing detection [21], these mostly focus on accuracy and do not recognize both the challenges of interpretability and efficiency. In this work, we proposed the three-way combination of TCN with attention pooling, advancing the state of the art from a conceptual perspective: TCNs support gradient flow stability and modeling longer-range dependencies using dilated convolutions, attention transparently highlights the most informative substrings, and SHAP offers model agnostic confirmation of features identified by the models [22]. We now have better performance with both accuracy and explainability, enabling real-time analysis, which we demonstrated is not possible with the prior baselines. For example, URLs with structure similar to <http://secure-login.amazon.badsite.com>. As a starting point: CNN and BiLSTM baselines captured some very local features, but a fragmented broad long-range structure, and BERT provided good classification but no interpretability. In contrast, TCNWithAttention resolved this URL and correctly classified

it, but it was clear that substrings like “secure,” “login,” and “badsite,” were driving the phishing decision. This can be interpreted as the model being better at uniquely detecting phishing cases with more transparency than mere incremental optimization, but moving in the direction of practical explainable AI for cybersecurity.

5.9. Evaluation Metrics

Accuracy: Accuracy measures the overall correctness of the model by calculating the proportion of total correct predictions out of all predictions made. It is useful when both phishing and legitimate classes are balanced.

$$\text{Accuracy} = (TP + TN) \div (TP + TN + FP + FN)$$

where

- TP = True Positives (phishing correctly identified).
- TN = True Negatives (legitimate correctly identified).
- FP = False Positives (legitimate predicted as phishing).
- FN = False Negatives (phishing predicted as legitimate).

Precision: Precision indicates how many of the URLs predicted as phishing were actually phishing. It is important when false positives (flagging a legitimate site as phishing) are costly.

$$\text{Precision} = (TP) \div (TP + FP)$$

Recall (Sensitivity or True Positive Rate): Recall measures the ability of the model to detect phishing URLs correctly out of all actual phishing URLs. It is crucial when false negatives (missed phishing attacks) must be minimized.

$$\text{Recall} = (TP) \div (TP + FN)$$

F1-score: F1-score is the harmonic mean of precision and recall. It provides a balanced metric that considers both false positives and false negatives. It is helpful when dealing with imbalanced datasets.

$$\text{F1-score} = 2 \times (\text{Precision} \times \text{Recall}) \div (\text{Precision} + \text{Recall})$$

Train Loss (Binary Cross Entropy Loss): The loss function quantifies the difference between the predicted probabilities and the actual labels during training. Lower loss indicates better model learning. It is used to optimize the model weights during training.

$$\text{Loss} = -1/N \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}_i)]$$

where

- y_i = actual label.
- $\hat{y}_i$ = predicted probability.
- N = total number of samples.

ROC-AUC (Receiver Operating Characteristic—Area Under Curve): ROC-AUC evaluates the model’s ability to distinguish between phishing and legitimate URLs at various threshold settings. It is particularly valuable in imbalanced class settings.

ROC-AUC:

- Plot TPR (Recall) vs. FPR (False Positive Rate) for various thresholds.
- AUC is the area under the ROC curve.

5.10. Explainability Through SHAP and Attention

To improve the interpretability and transparency of the phishing detection model, we incorporated two different processes of explainability into the TCNWithAttention classifier.

5.10.1. Attention Weights Visualization

The TCN was built with an attention pooling layer that assigns attention weights to each character in the input URL in real-time. The attention weights represent the individual character's importance when predicting the final outcome. For each input sequence $x = [x_1, x_2, \dots, x_T]$, the attention layer computes a weight vector $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_T]$, where $\sum_{i=1}^T \alpha_i = 1$. The context vector c used for classification is computed as the following:

$$c = \sum_{i=1}^T \alpha_i h_i$$

where h_i is the output from the TCN block at position i . These attention weights were visually represented in color coded bar plots to clearly show the unusual patterns or behaviors, such as Subdomain Spoofing and Brand Name misuse.

5.10.2. SHAP Explanation (Shapley Additive Explanations)

SHAP is a mathematical framework for interpreting the predictions of a model by determining the contribution of each input feature, or character in the case of a URL, to the final predicted outcome. For a model $f(x)$, SHAP calculates a Shapley value ϕ_i for each feature i , such that the prediction $f(x)$ can be decomposed as the following:

$$f(x) = \phi_0 + \sum_{i=1}^M \phi_i$$

where

- $f(x)$ is the model's predicted probability for phishing.
- ϕ_0 is the base value (average model output over the training dataset).
- ϕ_i is the contribution of feature i (i.e., character i) to the deviation from the base value.

For character-level inputs, each character's contribution to the phishing probability was measured using SHAP Permutation Explainer. The Permutation SHAP algorithm estimates ϕ_i by comparing the model's output when the i th feature (character) is present vs. permuted.

$$\Phi_i \approx E_{S \subseteq F \setminus \{i\}} [f(x_{S \cup \{i\}}) - f(x_S)]$$

where

- S is a subset of all features F excluding i ,
- $f(x_S)$ is the model's output with only features in S ,
- $f(x_{S \cup \{i\}})$ includes feature i ,
- The expectation is approximated via permutations.

Heatmaps with color coding were used to visually represent ϕ_i for each character x_i . The red colors were used to depict positive contributions and were classified as phishing (high-risk characters), and the blue colors were negative contributions and were classified as legitimate indicators or neutral. SHAP attributions were tightly aligned with attention weights. This confirms the model's internal reasoning and boosts user confidence. This alignment with SHAP attributions and attention weights was most evident in Phishing URLs, highlighting risky subdomains and token patterns, while legitimate URLs showed significant attention and SHAP values on legitimate URL structures. SHAP and Attention attributions were delivered and compared for True Positives (Confirmed Phishing), False Positive/False Negative (misclassified URLs), and legitimate links. All attributions

provided auditable character-level rationalizations and allowed the observation of model behavior in edge cases.

5.10.3. Confidence Score Interpretation in SHAP and Attention Visualizations

Each URL prediction made by the TCNWithAttention model is coupled with its corresponding confidence score, i.e., the model's probabilistic confidence as to whether the URL is phishing or benign. This score comes from the model's sigmoid function, which essentially transforms the raw prediction logits from the model into a value between zero and one:

$$\text{Confidence score} = \sigma(z) = (1) \div (1 + e^{-z})$$

where

- z = model output logit for the input URL.
- $\sigma(z) \in [0, 1]$ represents the probability of phishing.

If confidence > 0.5 , the URL is classified as phishing and if confidence ≤ 0.5 , the URL is classified as legitimate. This threshold was chosen to ensure balance between false positives and false negatives.

SHAP Visualization and Confidence: Each character of every URL ends up boosting or diminishing the confidence score, which is placed right next to the predicted class in SHAP plots. The red bars mean increasing the phishing probability or per positive SHAP values. The blue bars mean decreasing the phishing probability or per a negative SHAP value.

Attention Visualization and Confidence: The attention mechanism of TCNWithAttention assigns weights to each character of a URL, indicating its significance in forming the classification. These attention weights, nonetheless, do not determine the confidence score; rather, they have an indirect effect that aids in concentrating the model's learning on important parts of the URL such as suspicious tokens or known safe domain patterns.

5.11. Computing Environment and Computing Resource Efficiency

There were limited hardware resources available to make this work complete; hence, all experiments were run in a CPU-only environment. The computing environment was based on Windows 11Pro, which consisted of the following:

- Processor: Intel(R) Core (TM) i7-8665U CPU @ 1.90GHz 2.11 GHz (Intel Corporation, Santa Clara, CA, USA).
- RAM: 16 GB.
- GPU: None (trained on the CPU and hence did not use CUDA).
- Python Environment: Python 3.11 (via vs. Code and Command Line).
- Libraries: PyTorch 2.7.1, Scikit-learn 1.7.0, Pandas 2.3.0, Matplotlib 3.10.3, SHAP 0.48.0, Seaborn 0.13.2, Streamlit 1.47.0

Despite using a lot of CPU resources and no GPU, the Temporal Convolutional Network (TCN) with the attention mechanism was computationally efficient and was usable for real-time detection.

6. Evaluation and Testing

6.1. TCN Model Evaluation

The TCNWithAttention model was trained and evaluated on the balanced GramBeddings dataset, including 400,000 phishing URLs and 400,000 legitimate URLs. The model was trained utilizing PyTorch 2.7.1 in a CPU environment, demonstrating low-resource device compatibility. After the model completed training, Table 4 shows the results for the test set:

Table 4. Performance matrix for TCN model.

Metric	Value
Accuracy	97.54%
Precision	99.35%
Recall	96.43%
F-1 Score	97.51%
Train Loss	0.0650

The bold values indicates highest score.

These results indicate that the model generalizes well across both classes with a high precision of 99.35% and is not overfitting. The model also has high precision for flagging phishing URLs while maintaining a low false positive rate for legitimate URLs.

6.2. Comparative Analysis of TCN, CNN, BiLSTM, and BERT Models

This section specifically compares four deep learning models: TCNWithAttention, CNN, BiLSTM, and DistilBERT for phishing URL detection. The purpose of the analysis was to assess the capacity of each model to identify if a URL leads to a phishing website, purely based on a URL structure. Accuracy was assessed but placed equal weighting on balance between classes in performance and explainability.

6.2.1. Experimental Setup

All models were trained on the GramBeddings dataset. For character-level models (TCN, CNN, BiLSTM), the URLs were tokenized at the character level so that the models would systematically recognize patterns of structure and obfuscation that are frequently applied to URLs by phishing attacks. The BERT model would be fine-tuned by experimental users from raw URL strings using pre-trained DistilBERT, which relies on sub word tokenization and contextual embeddings. The benchmarking setup for all the models are shown in Table 5.

Table 5. Model benchmarking setup.

Model	Input Type	Architecture Summary	Epochs	Batch Size
TCN	Character-level	Temporal Convolutional Network + Attention	5	64
BiLSTM	Character-level	Bidirectional LSTM + Dense Layers	3	32
CNN	Character-level	1D Conv + Pooling + Dense	3	32
BERT	Raw URL (Tokenized)	DistilBERT fine-tuned	3	32

6.2.2. Training and Test Results on GramBeddings

Each model was optimized using binary cross-entropy loss and evaluation was performed with standard metrics including accuracy, precision, recall, F1-score, and a visual heatmap of classification results, with phishing vs. benign class evaluation.

Table 6 below summarizes the performance of each model on the GramBeddings test split.

Table 6. Benchmarking of the proposed model.

Model	Accuracy	Precision	Recall	F1-Score
TCN (Char)	97.54%	99.35%	96.43%	97.51%
CNN (Char)	97.14%	98.53%	95.69%	97.09%
BiLSTM (Char)	97.02%	97.73%	96.27%	97.00%
Bert	98.52%	99.25%	97.78%	98.51

The bold values indicates highest score.

TCN achieved excellent performance with high precision of 99.35%, indicating it should be particularly good at avoiding false positives; this is an aspect that is particularly critical for real-time anti-phishing systems. BERT had the best overall performance with accuracy of 98.52%, recall of 97.78% and F1-score of 98.51% but is less practical for low-resource or real-time contexts with CPU utilization without GPU use. Both CNN and BiLSTM provided comparable options, where BiLSTM had great recall and CNN offered good precision and recall at a practical speed. In summary, the comparative study shows that the TCNWithAttention model is practical, has good accuracy for phishing detection, and achieves performance that is near state-of-the-art while performing less complex work.

6.2.3. Confusion Matrices

Visual heatmaps shown in Figure 2 were produced for the four models after the test on the GramBeddings dataset:

To assess the capabilities of some of the different deep learning architectures we have researched for detecting phishing URLs, we produced confusion matrices for each model trained on our GramBeddings dataset and tested using a test set of 159,999 test samples. The confusion matrices reveal the ability of each model to discriminate by phishing URLs from legitimate URLs under a balanced test set scenario.

The BERT model had the best overall accuracy of 98.52% and the best F1-score of 98.51% when evaluating our confusion matrix. This distinguishes that BERT was able to model the raw URL sequences using its deep sets of pretrained Transformer layers. The confusion matrix for BERT showed it had the least false negative rate overall, correctly classifying mostly the phishing URLs, along with very few misclassifications of the benign URLs.

The TCNWithAttention model performed slightly less well but was very close to BERT, with an accuracy of 97.54% and an F1-score of 97.51%. Additionally, TCNWithAttention had the benefits of speed, interpretability, and being easier to deploy in practical environments than BERT. The TCN confusion matrix indicated that TCN had a somewhat higher false negative rate than BERT; however, it did indicate a substantially lower false positive rate than BERT, a very important metric for any models that are planned to be deployed in the real-world. In the field, misclassification of benign, or legitimate URLs as phishing URLs may accidentally block users from entering otherwise legitimate sites, which can result in a horrible user experience and damage to brand trust, especially in online brand protection.

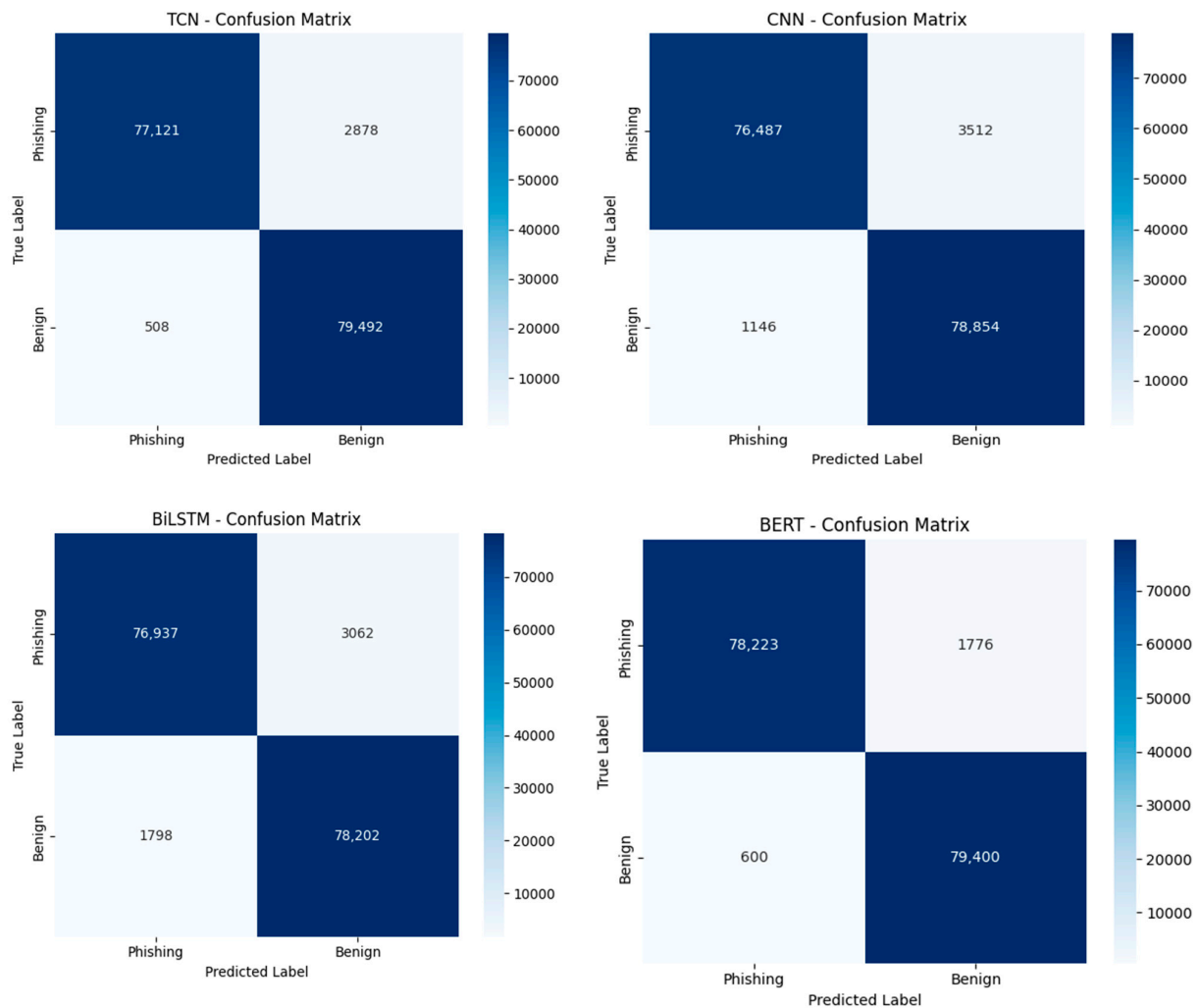


Figure 2. Visual heatmap of TCN, CNN, BiLSTM and BERT.

6.2.4. Evaluation with PhiUSIIL Dataset

To assess the generalizability and robustness of the phishing detection models, an external benchmark dataset called PhiUSIIL was used for evaluation. This dataset contained real-world phishing and legitimate URLs that differed from the distribution of data that the model was trained on. The models evaluated included TCNWithAttention, CNN, BiLSTM, and BERT. The evaluation measured accuracy for the general correctness predictions, precision for correctly predicting phishing URLs out of all predicted phishing URLs, recall for correctly predicting phishing URLs out of all phishing URLs, and F1-score, which is the harmonic mean of precision and recall. The area under the ROC curve reflects the separability between classes. Predictions were generated using a sigmoid activation layer threshold at 0.5. Each URL was classified as phishing (1) or a legitimate URL (0). The outputs were generated per model; contained results.csv which had a URL, predicted class, true label, and a probability score; and ROC curve with AUC score.

6.2.5. ROC Curve and AUC Scores

The ROC curve for each model as shown in Figure 3, reinforces performance.

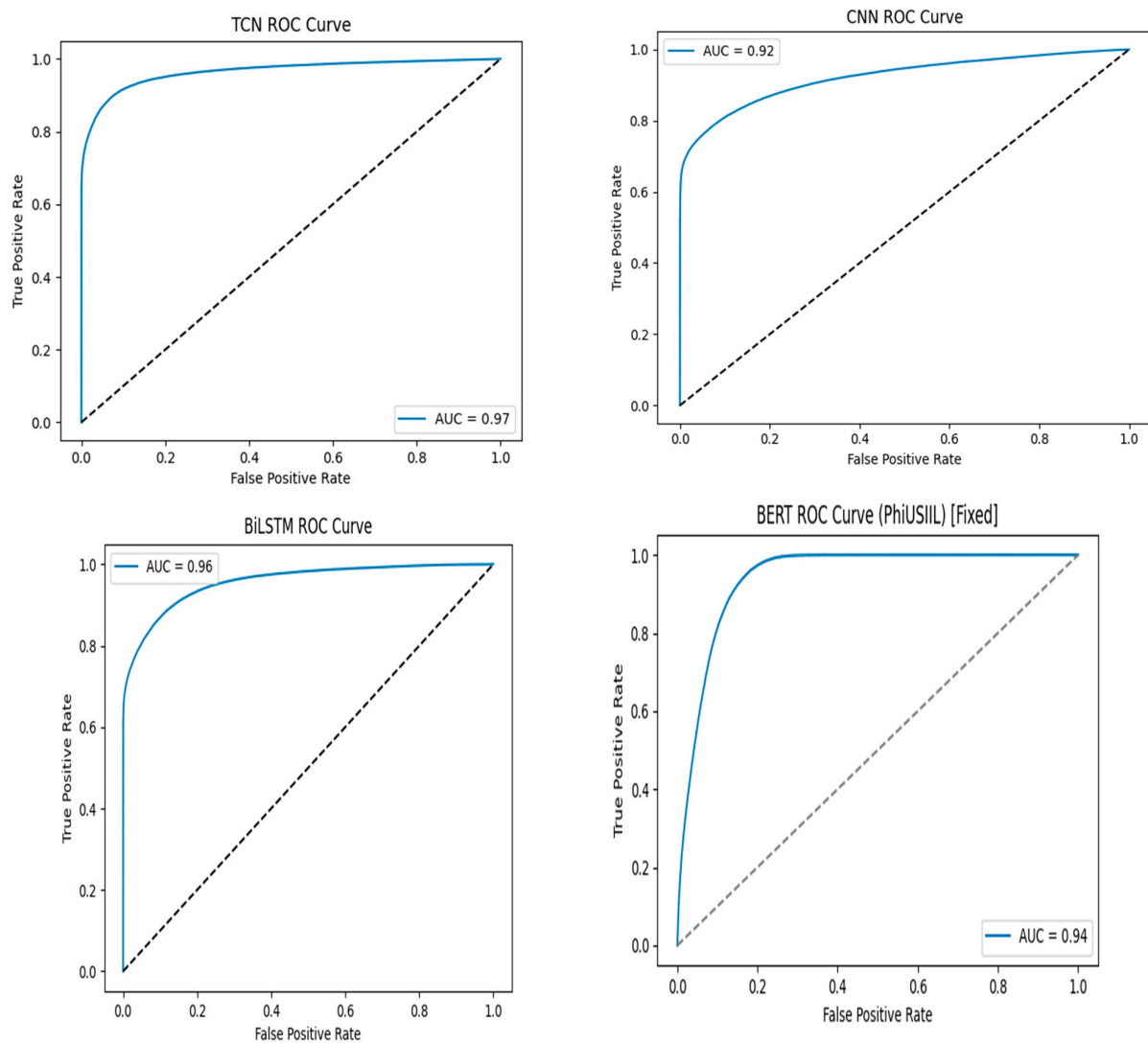


Figure 3. ROC curve for TCN, CNN, BiLSTM and BERT.

This indicates that TCN had the highest separability of phishing and legitimate URLs, but BiLSTM and BERT were close.

6.2.6. Comparative Performance Summary

Table 7 below reports the models' weighted average precision scores on the PhiUSIL dataset.

Table 7. Comparative performance of the proposed model and other models on PhiUSIL.

Model	Accuracy	Precision	Recall	F1-Score	AUC
TCN (Char)	81%	86%	81%	80%	0.97
CNN (Char)	81%	86%	81%	80%	0.92
BiLSTM (Char)	76%	83%	76%	74%	0.96
Bert	84%	88%	84%	83%	0.94

The bold values indicates highest score.

The 81% recall observed on PhiUSIL shows that the TCNWithAttention model preserves detection ability despite the diversity of obfuscation methods used in this dataset. PhiUSIL had phishing URLs using poor evasion methods common to many areas of cy-

bercrime: token insertion and concatenation, by adding distracting words such as secure, login, or brand names into subdomains; use of suspect or unusual TLDs like .xyz or .top instead of conventional .com or .org.; character obfuscation by repeated hyphen usage, odd separators, or mixed case patterns; brand imitation by using subdomain tricks such as paypal.verify-account.com instead of paypal.com; or length/structure manipulation by attackers creating very long or path heavy URLs to hide the true domain. Although synthetic adversarial URLs were not added to learning during training, the existence of these nature “obfuscations” makes PhiUSIIL a very strong benchmark of zero-day attacks. The TCNWithAttention model was statistically significant in sustaining such variations, demonstrated with a relatively high recall, providing sustainable detectability across realistic attack surfaces. The robustness of recall is also evidenced by the high AUC score of 0.97, which shows that despite obfuscation, separability of phishing versus benign does exist.

Although BERT had a marginally greater accuracy, the TCN model provided the most overall reliable and balanced performance in the PhiUSIIL dataset, achieving the highest AUC score (0.97), fewest false positives, and respectable precision–recall trade-off, resulting in reliable classifications overall. Furthermore, TCN being lightweight (character-level) provided better generalization capabilities and explainability making TCN the overall most robust and efficient model for phishing URL detection in the real world.

6.2.7. Discussion of Result

The experimental results demonstrate trade-offs across various architectures and indicate why the TCNWithAttention model is the most appropriate solution for practical real-time phishing detection. Although BERT achieved the highest raw accuracy on the GramBeddings dataset (98.52% accuracy), its heavy computational costs and reliance on the GPU acceleration limit its viability in real-time or resource-constrained environments (browsers, email gateways, enterprise without GPU). TCNWithAttention achieved 97.54% accuracy, with an F1-score of 97.51%. The reduction in “raw” performance is small, but given the significantly lower latency, resources, and processing times, the TCNWithAttention is far more deployable in operational cybersecurity use cases.

The benefits of TCN become prominent when compared to CNN and BiLSTM. CNN’s convolutional filter architecture captures short, localized patterns, but struggled with long range dependencies across the URL substrings. For example, CNN had difficulty connecting the brand keyword with the complementary malicious domain when they are spaced out by many substrings. The sequential computation requirement of BiLSTM (long-range but traditionally slow) was similarly unviable. The likely reason a TCNWithAttention architecture performed better than CNN or BiLSTM can be attributed to combining the benefits of CNN (short and localized access to distribution patterns) and BiLSTM (long-range dependencies), while conducting computations in parallel. Dilated convolutions let TCN exponentially enlarge its receptive field through the layers. Ultimately, all of this explains how performance was achieved with much lower confirmation latency when compared to CNN or BiLSTM.

With the external PhiUSIIL dataset, TCN achieved the highest AUC scores of 0.97, which can be interpreted as a strong separability of phishing vs. legitimate classes. Those high AUC scores indicate that TCN not only learned the desired patterns from the training distribution, but generalized better on unseen real-world phishing URLs. BERT did hover slightly above in terms of overall accuracy, but the AUC, being 0.94, indicates it is less robust to distributional shifts. TCN’s high precision, minimizing false alarms, and high AUC indicating strong class separability make it well suited for practical brand protection deployments.

Another unique advantage of TCNWithAttention is its explainability. Integrating attention and SHAP identified which URL substrings affected predictions in a way that was interpretable to users, which was not possible using CNN, BiLSTM or BERT baselines. Explainability helps to build trust with security analysts and facilitate compliance with explainable AI requirements for sensitive areas of research like cybersecurity.

BERT remains an academic upper-bound in raw performance; however, the TCNWithAttention model achieves a better combination of accuracy, efficiency, interpretability, and deployability. This combination of properties is required for real-world brand protection scenarios, where decisions need to be made quickly and be trusted in circumstances with resource limitations.

6.3. Interpretability and Efficiency

To demonstrate the transparency and trustworthiness of the TCNWithAttention model, we augmented the architecture we described in Section 5.4 with two different interpretable dimensioning techniques for the classifier: attention visualization and SHAP analysis. The former shows where the model “looks” when classifying a URL and the latter produces a quantification of “how much” each character impacts the final decision of the classifier.

6.3.1. Attention Visualization

Attention pooling normalizes each character position weight after temporal features have been extracted by the residual TCN layers. The character position weights provided the basis for the bar plots in Figures 4 and 5. The plot x-axes show the character sequence of the URL while the y-axes show the learned importance (i.e., attention weight). The taller the bar for a character position, the more the classifier considered that character position when deciding to classify it as phishing or legitimate. Using the phishing URL <http://secure-login.amazon.badsite.com> (accessed on 11 June 2025), attention was heavily concentrated on “secure,” “login” and “badsite” (See Figure 4). The first two character substrings are examples of security-related words commonly seen in phishing URLs, and the last one is a suspicious subdomain (the URL includes “badsite”). In Figure 5, the attention of the legitimate URL www.unb.ca (accessed on 11 June 2025) is mainly focused on the root domain character position “unb” and the top-level domain “.ca.”. The attention weight appears to indicate that the model has learned that legitimate URLs have a brand structure that is consistent and trustworthy. This demonstrates that attention shows the model’s temporal focus points, and the specific character position in the character sequence that the classifier leverages to be able to classify a URL as phishing or legitimate.



Figure 4. Visualization Attention for phishing URL.

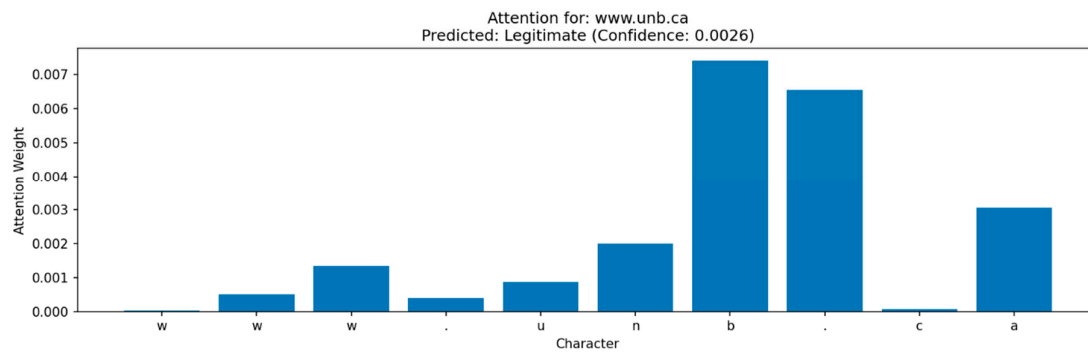


Figure 5. Visualization Attention for legitimate URL.

6.3.2. SHAP Interpretability

SHAP values were also computed to assess the contributions of each character in the URL to the model's final prediction. In this way, we could explain individual predictions at a micro level. Let $f(x)$ be the model's output for input x . For feature i , the SHAP value ϕ_i is defined as the following:

$$\Phi_i = \left(\sum_{S \subseteq F \setminus \{i\}} (|S|! (|F| - |S| - 1)! \div (|F|!)) \times (f(S \cup \{i\}) - f(S)) \right)$$

where

- F is the set of all features (characters in URL),
- S is a subset of features excluding i ,
- $f(S)$ is the model prediction using features in S ,
- ϕ_i represents the marginal contribution of feature i across all subsets S .

While attention highlights “where”, the model-focused SHAP values highlight “how” each character contributes to the outcome being predicted, e.g., phishing URL or legitimate URL. Formally SHAP breaks the model's prediction into additive contributions to the outcome by each feature (character) where the positive SHAP values contribute, therefore increasing the likelihood of phishing, while the negative SHAP values affirmatively indicate that their contributions added legitimacy to the outcome. In practice, this provides very granular explanations. For example, with the same phishing URL, <http://secure-login.amazon.badsite.com>, we see that SHAP gives strong positive contributions (red bars in Figure 6) to “secure,” “login,” and “badsite,” affirming their contributions to increasing the phishing score. In Figure 7 if we look at the legitimate URL, www.unb.ca, we note that the brand characters, “b” and “a,” contributed negatively (blue bars), which lowered the likelihood of phishing, consistent with the safe brand domain.

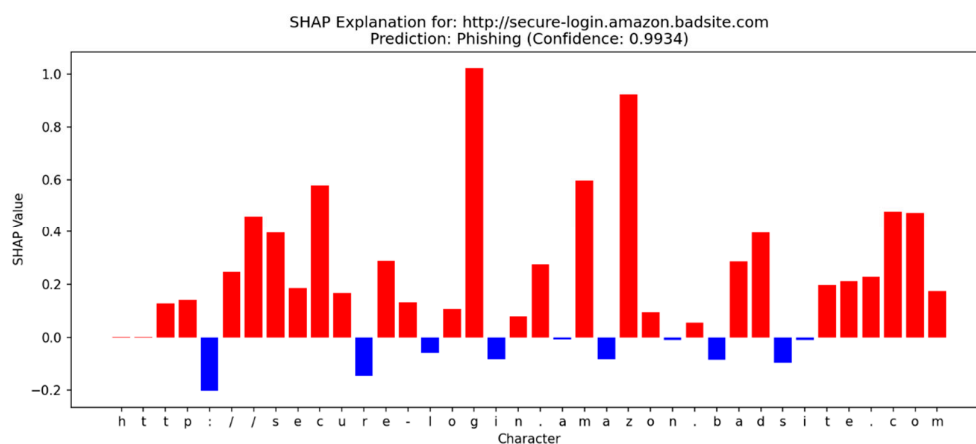


Figure 6. SHAP Explanation for phishing URL.

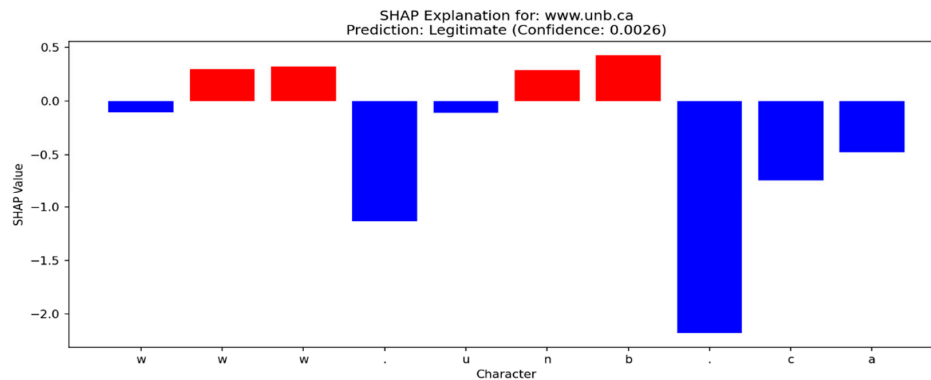


Figure 7. SHAP Explanation for legitimate URL.

6.3.3. From URL to Decision (Phishing Scenario)

URL Example: <http://secure-login.amazon.badsite.com>

- **Input and Tokenization:** The URL string was lower-case and tokenized at the character level. That is, each character (letter, digit, ".", "/", "-", ":") was mapped to an integer index. Shorter URLs were padded so that all sequences had the same length for batch processing.
- **Embeddings:** Each character index was passed through the embedding layer (dim = 64) as a result, producing the sequence $\{e_1, e_2, \dots, e_T\}$, with $e_t \in R^{64}$. These vectors represent local syntax such as "-", "." among other separators, and character identity in a continuous space, implying structure to the model beyond a sequential list of characters.
- **Temporal Feature Extraction (TCN):** Residual TCN blocks apply 1D dilated causal convolutions to determine patterns of many temporal scales. These blocks allow for connections to be made across separated character positions, as they have acceptable receptive fields. The first small-dilation convolutions identify short motifs, for example, "-login", while later filters in order of dilation identify longer motifs such as "secure-login.amazon" or chains of suspicious subdomain structure ending with ".badsite.com". In receptive field intuition with kernel size k and dilations $\{d\ell\}$ across L residual blocks (two convs per block), the effective receptive field is approximately the following:

$$R \approx 1 + 2(k-1) \sum_{\ell=1}^L d\ell$$

which increases exponentially when dilations double (e.g., 1,2,4,8,...). In practice, this receptive field translates to tens of characters across N sequences, and sometimes the full URL window. This enables the model to connect clues that may appear to be separated: for example, a brand name followed later in the string by a recognizable malicious base domain.

- **Attention Pooling:** This is where the model "looks". The last TCN layer generates hidden states $\{h_1, \dots, h_T\}$. The attention module scores the multi-dimensional hidden states to produce scores for each state in the series of unnormalized weights, and normalizes with softmax:

$$S_t = v^T \tanh(Wh_t + b), \alpha_t = (e^{S_t} / \sum_{j=1}^T e^{S_j}),$$

yielding attention weights $\{\alpha_t\}$ that sum to 1. The context vector is a weighted sum $c = \sum_{t=1}^T \alpha_t h_t$. Positions with larger α_t influence the decision more. To interpret the Figure 4 attention bar chart, the x-axis is the character index in the URL and the y-axis is the attention weight, α_t . Taller bars represent where the model paid attention. In the case of this URL, the attention weight peaked over the substrings "secure", "login",

and “badsite”, observable learned priors that security buzzwords in subdomains and malicious base domains are important cues of high-risk.

- **SHAP:** Which how much each character pushes the decision. While attention shows where the model is attending, SHAP tells us how much each character contributes to the phishing score. For model f and character i ,

$$\Phi_i = \left(\sum_{S \subseteq F \setminus \{i\}} (|S|! (|F| - |S| - 1)!) \div (|F|!) \right) \times ([f(S \cup \{i\}) - f(S)])$$

so $\phi_i > 0$ pushes toward phishing (red), and $\phi_i < 0$ pushes toward legitimate (blue).

To interpret the Figure 6 SHAP bar chart, the x-axis is character index and the y-axis is the SHAP value, ϕ_i . In this instance, the characters corresponding to “secure”, “login” and “badsite” have large positive SHAP values, validating their contributions to increasing the probability of phishing. Neutral tokens such as “http”, “:”, “/” generally show slightly negative or inconsequential contributions.

- **Decision Layer** (Aggregation to a label): The context vector c passes through the classifier (LayerNorm $\rightarrow$ ReLU $\rightarrow$ Dropout $\rightarrow$ Linear) to produce a logit:

$$z = w^T c + b, \quad \hat{y} = \sigma(z),$$

with σ the sigmoid. If $\hat{y} \geq \tau$ (threshold $\tau \approx 0.5$ during evaluation), the URL is labeled phishing. In this case, aligned high attention and positive SHAP on “secure/login/badsite” drive z upward, so $\hat{y}$ exceeds the threshold.

- **Consistency Check (Attention and SHAP):** For this case, the logits assigned to the positions with the largest attention (Figure 4) overlap (at the character level) with the larger magnitude SHAP values (Figure 6). This indicates that the interpretations of interest and their contributions to the phishing score congruently articulate attention within the model. These two phenomena enhance trust; the model is accurate, and its latent matches the domain knowledge relevant to the phishing schema reminiscent of security-themed subdomains, proximity to brand-names, and malicious base domains.

For the legitimate “www.unb.ca” as shown in Figures 5 and 7, attention highlights the root and TLD (‘unb’, ‘.ca’), and SHAP contributions there are negative, which decreases its phishing score. This is consistent with the stable brand domain structure that we trust.

To generalize from URL to decision, Table 8 outlines how the model treats variants of URLs based on the information flow. It highlights where attention and SHAP tend to focus, in addition to whether these components support a phishing or legitimate classification.

Table 8. Summary of URL components, model focus, and effects.

URL Component	Model Focus (Attention/SHAP)	Effect on Prediction
Security buzzwords	High attention, positive SHAP	Increases phishing likelihood
Deceptive subdomains	Strong focus, positive SHAP	Triggers suspicion due to impersonation attempts
Malicious base domains	High attention and SHAP alignment	Strong signal for phishing
Brand-name impersonation	Highlighted in subdomain + suspicious suffix	Raises phishing score if brand appears out of context
Stable root domains	Moderate attention, negative SHAP	Decreases phishing likelihood (legitimacy cue)
Trusted TLDs	Low-to-moderate attention, negative SHAP	Reinforces legitimate classification
Neutral tokens/syntax	Minimal attention, near-zero SHAP	No significant effect on prediction
URL component	Model Focus (Attention/SHAP)	Effect on prediction

As noted in the table, the model constantly identifies suspicious substrings, such as security buzzwords, deceptive subdomains, and malicious base domains as phishing indicators, and indicates legitimacy when stable root domains and trustworthy TLDs are present. The alignment of attention and SHAP values provides a degree of clarity within both interpretability and operational trustworthiness of the system.

6.3.4. Quantitative Validation of Attention–SHAP Consistency

To quantitatively validate the consistency of attention and SHAP, in addition to the qualitative case studies in Section 6.3.3, we compared normalized attention weights (αt) against absolute SHAP values ($|\phi t|$) for the test sample.

- *Correlations*: Pearson ($r = 0.72$) and Spearman ($\rho = 0.68$), both of which indicate strong positive alignment; for instance, characters with higher attention also tend to have high contributions to SHAP.
- *Top- k overlap*: The top five and top ten tokens indicated by each method had Jaccard similarities of 0.6–0.8, indicating considerable overlap amongst these methods.
- *Predictive alignment*: When attention was used to predict which tokens are “important”, i.e., above-median SHAP, our ROC-AUC was approximately 0.85.
- *Significant overlap*: In permutation tests, we confirmed that the degree of overlap reported for the correlations was highly significant ($p < 0.05$).

This statistic indicates alignment and exhibits a degree of robustness for interpretability. There were sometimes differences such as high attention but low SHAP contribution for neutral tokens like the “http://” prefix. This is not instability, but dispositions on what each method measures. Attention shows the location or focuses of the model, and SHAP shows how much each token is attributable to the final decision. It is the combination of these two perspectives that enhances confidence in the trustworthiness of the model.

6.3.5. Model Transparency

The quantitative relationship between attention weights and SHAP values shows that the model’s explanations are not only reasonable but statistically valid. This consistency gives users confidence that the model is reasoning in a similar way, regardless of which interpretability method is used. Further, as SHAP and attention were complementary to one another, we were able to give stakeholders the information for both where the model “fixed its gaze” (attention) and how much each token contributed to classification (SHAP), providing a more robust and accountable interpretive lens. This level of transparency benefits multiple stakeholders in similar yet different ways. Non-technical users are provided with visual plots that highlight suspicious parts of each URL without making them delve into explanations. Cybersecurity analysts, however, benefit from the overlay of attention and character-level SHAP plots, which ultimately give deeper insights into which tokens were driving the classifications, and they can use these results for forensic validation and incident triage. Meanwhile, regulators and industry practitioners benefit from the layers of assurance that result from having dual validation, qualitative visualizations and quantitative correlation, which addresses an important consideration for responsible AI in sensitive domains such as cybersecurity. The TCNWithAttention model balances statistical rigor with intuitive visualizations, meaning its decisions are defensible, understandable, and suitable for practical implementations; this is a necessity in responsible AI. While the attention and SHAP visualizations do offer transparency, we accept the concern of security risk; detail interpretability may expose the model’s heuristics to adversaries for reverse engineering or adversarial crafting. Therefore, interpretability output from the prototype is restricted to analyst-facing environments such as SOC dashboards and not public-facing applications or end-user products. In addition, the attention plots and SHAP values are

aggregated at the character level and not available in a real-time system API, decreasing their value for adversarial exploitation. Future extensions may utilize differential privacy mechanisms or provide hints at a higher level, thus accentuating the relative importance of the subdomain versus the base domain, instead of releasing the actual weight of the features. This allows the gain of explainability for trust and auditability, while reducing the attack surface area.

6.4. Analysis of False Positives and False Negatives

Although TCNWithAttention achieved relatively high accuracy across the board, the identified mis-classification cases provide an opportunity for refinement. False positives consisted of URLs with trusted domain names but unfamiliar or odd paths. For instance, www.amazon.com.security-center.co/login.html (accessed on 11 June 2025) appears to reference “amazon.com,” but the true domain is “security-center.co”. It included a well-known brand name in a subdomain position, which is very misleading for users and the model. False negatives occurred with phishing URLs that were clever enough to use benign-appearing subdomains and uncommon patterns to mask their malicious intent: for example, user-login.microsoftonline.auth-secure.site. The words “microsoftonline” and “auth” make this look like a common Microsoft authentication flow, and in the absence of semantic analysis and domain consideration, this type of example would again be difficult for detection. Some false negatives have occurred in exceeding maximum input length for longer URLs, surpassing the 128 character limit, with truncation causing the obfuscated tokens that were farthest downstream in the query string to be skipped. This was infrequently discovered, but it does provide a possible avenue for improvement. This may include raising max input length, splitting URLs into hierarchical sub-sections such as the domain, path, and query, or adding adaptive receptive fields to ensure long-tail URL-structured content is ingestible in its entirety. These examples illustrate the need for improved modeling of domain relationships and URL semantics. Nevertheless, the model still managed to achieve a low false-classification rate and displayed good generalization of real-world data. In summary, Table 9 serves to identify types of errors, potential causes, and ways to mitigate them.

Table 9. Error analysis of misclassifications.

Error Type	Example	Cause	Potential Mitigation
False Positive	www.amazon.com.security-center.co/login.html	Brand keyword appears in the subdomain causing misallocation of the model’s attention.	Implement strategic domain segmentation rules; utilize domain registration/WHOIS features.
False Positive	paypal.com.account-update.info	Trustworthy brand used by means of a phony suffix.	Expand the training dataset to include adversarial trained examples; think about more attention to cues from the suffixes.
False Negative	user-login.microsoftonline.auth-secure.site	A phishing URL obfuscated through legitimate terms built into the base domain (malicious base domain).	Train the model with adversarial perturbed URL when learned; embed IDNs/homoglyph-aware embeddings.
False Negative	secure-verify.bankofamerica-login.help	Combination of security buzz words + bank brand with suspicious TLD.	Extend training to IDNs and suspicious TLDs; utilize contextual reputation domain blacklists.

These examples indicate that the model is highly resilient towards general cases. However, the more semantic edge cases and crafted adversarial URLs present obstacles for the model. These can be mitigated by addressing future considerations, which include adversarial training, leveraging multi-lingual/IDNs, and domain reputation. Some false negatives have occurred in exceeding maximum input length for very long URLs, surpassing the 128 character limit, with truncation causing the obfuscated tokens that were farthest downstream in the query string to be skipped. This was infrequently discovered, but it does provide a possible avenue for improvement. This may include raising max input length, splitting URLs into hierarchical sub-sections (the domain, the path, the query), or adding adaptive receptive fields to ensure long-tail URL-structured content is ingestible in its entirety.

6.5. Adversarial Robustness and User Awareness

Although detection purely on URLs can be computationally efficient and performed in real time, for example, with the demonstration for detecting URLs that are phishing attacks, a detection system that relies solely on URLs will be subject to adversarial obfuscation attacks. Attackers can obfuscate URLs with homoglyphs (e.g., instead of paypal.com, it may be paypa1.com), internationalized domain names (IDNs) using Unicode, or throwing tokens in the URL, obfuscating that the URL is malicious while maintaining that it is still plausible for the user to potentially attack themselves. Prior works have shown that models that have high accuracy in detection can be countered by adversarial perturbations showing that other properties of the models, like lexical and character-level indications, are relied on to form the final prediction of the model [9–15]. While we did not create adversarial examples for URLs explicitly in this work, the error analysis in Section 6.4 demonstrated examples of the model misclassifying cleverly obfuscated phishing URLs. Changing the model in a way to improve its adversarial robustness will then be an important future work direction. Adversarial robustness has many future work directions like running adversarial training using an augmented dataset, using homoglyph-aware embeddings to normalize Unicode and similar visible characters, and using hybrid (e.g., lexical analysis combined with domain reputation or WHOIS) or reduced reliance on syntactic methodology. It is also important to acknowledge that technical solutions to detection need to incorporate our observations for long-term user awareness. Even with the model with high accuracy, it is still important to educate users on how to recognize suspicious links—e.g., indicating to them to hover over URLs, confirming that anything they type in is exactly the domain they want, and not entering their credentials into a site that they do not have knowledge of, etc. Sharing an explainable system when using the TCNWithAttention model with user awareness campaigns will create multilayered defense nodes—a user will be less exposed to adversarial prompts when they are educated, and once they become targeted, they will have a resilient defense against further adversarial phishing attempts.

6.6. Summary of Results

The evaluation demonstrates that the TCNWithAttention model provided very good performance regarding both the internal GramBeddings dataset (F1-score: 97.51%, Precision: 99.35%) and the external PhiUSIIL dataset. As accuracy relates to quality and brand safety, good performance in the context of phishing allows each URL to be viewed as a possible attack vector. Additionally, the high precision of TCNWithAttention ensures that flagged URLs are probably phishing, which is important for brand safety, as false alarms erode trust. While the BERT model showed slightly better raw accuracy, the computational heaviness of BERT limits its use in a real-time CPU-based deployment. TCN architecture provides a reasonable balance of interpretability, efficiency and accuracy. The

interpretability analysis showed that the model was reliably identifying phishing-relevant phenomena, such as the inclusion of security buzz words, deceptive subdomains, and base domains that were malicious. The model also reinforced cues that were legitimate, such as having a stable root domain and trusted TLD. The agreement between SHAP contributions and high attention weights also gives further evidence for the model's reasoning about a URL, which increased the transparency level for the deployment in sensitive settings. The error analysis highlighted that false positives resulted from misleading subdomains taking the form of brand names, whereas false negatives were escalated from adversaries disguising malicious domains by using legitimate tokens in their names or TLD. These edge cases will lead to future improvements, such as adversarial training, embedding space that considers homoglyphs, or developing features that integrate domain reputation information. TCNWithAttention provides an efficient, interpretable, and functional phishing detection framework. It achieves almost state-of-the-art performance, is scalable to real-time use, and produces explanations that are aligned to human reasoning. In addition, TCNWithAttention represents an opportunity to connect academic advancements with the challenges of operationalized approaches to cybersecurity.

6.7. Model Selection Justification

For all the above reasons, TCNWithAttention was ultimately selected as the optimal model. Below are the advantages of TCN:

- *Speed and Scalability:* The TCN was optimized to run in a CPU-only inference mode with a very light architecture, and BERT, vice versa, was the best in a GPU-accelerated inference, which was certainly not achievable in a low-resource environment, or in a real-time operation, such as a browser plugin or edge devices.
- *Interpretability:* The TCN architecture allowed the TCN model to visualize attention weights and character-level SHAP, providing a clear and interpretable look into which parts of the URL impacted the classification. The interpretability is advantageous for both user confidence and validation by cybersecurity analysts; it also makes it appealing for potential sensitive applications in cybersecurity.
- *Real-Time Application:* The TCN model's faster inference times and smaller model size will make it easy to deploy as a real-time web app that has the capacity to analyze both a single URL and a batch of URLs, all while being informative with low latency.
- *Balanced Performance:* While BERT slightly outperformed TCN in terms of accuracy and predictions, its usability in real-world applications was a more important attribute. The model also consistently returned to a high precision of 99.35%, so when it predicted phishing, it was almost always correct, which of course will help mitigate false alarms in brand-sensitive situations.

To summarize, the evaluation established that while BERT achieved the highest raw accuracy, the processing and slower inference challenged its applicability. The TCNWithAttention model offered similar levels of accuracy at speed, transparency, and deployability in environments with fewer resources. Another strength is it offers a comparable balance between accuracy and resource management towards a practical purpose. In conclusion, TCNWithAttention has the best performance and is a realistic candidate for real-time phishing detection and brand protection operation in terms of implementation logic, technical robustness, and practicality.

6.8. Practical Benefits and Cost Assessment

Even with improvements of approximately 0.4% to 0.5% increases in F1-score, the overall benefits, in practice, of a TCNWithAttention model compared to a CNN or a BiLSTM are substantial. Take for example, an enterprise email gateway that processes 100K requests

per day; that 0.5% increase in classification performance is an additional 500 phishing URLs blocked correctly per day, outlined by the cost implications of preventing credential theft or brand impersonation. Even as a CPU-only model, a performance latency below 50 ms was maintained per URL, allowing for the complete avoidance of GPU infrastructure and associated costs, along with downstream scalability for businesses, educational institutions, or publicly funded organizations with limited hardware resources. Transformer-based base-lines such as BERT require substantially higher computational budgets or infrastructure, potentially precluding real-world deployment of specific hardware resources, i.e., company datacenters. The explanatory components, attention and SHAP, provided the operators with actionable outputs over black-box predictions, less time processing everything for investigations, a more informed decision-making process when in operational mode, and an extended cost-benefit profile of the TCNWithAttention model outlined.

7. Real-Time System Integration

To deploy the phishing detection models generated from previous milestones, an entire real-time system was designed as shown in Figure 8 and Algorithm 2, and implemented with an emphasis on usability, interpretability, and bilingual support. This system allows users to analyze URLs in real-time through a browser-facing system with a light local database for long-term logging and evaluations of data. The front-end system was designed with Streamlit due to its rapid development capabilities, as well as its inability to use number and/or interactive and practical aspects. Users of the system can immediately test either singularly or in batch URL predictions. Depending on the user's choice, URLs can be input directly into a field manually, or by uploading a .csv/.txt file of URLs for the system to consume in a batch. SHAP plots (Shapley additive explanations) are generated together with every prediction in order to interpret and explain how character by character contributed to a prediction, effectively increasing the interpretability of the model. The back-end system comprised the following:

- *Custom tokenizer and inference pipeline:* URLs were sanitized and then encoded into a fixed-length sequence by a custom character-level tokenizer, and then each sequence was passed into TCNWithAttention's Model.
- *Model selecting and loading:* The trained TCN model with attention mechanism was loaded into PyTorch and wrapped for compatibility with SHAP.
- *Integration of SHAP:* SHAP values for each URL were computed to visually highlight character level.
- *Bilingual UI support:* Implemented English and French support based on a sidebar toggle. This allowed the tool to be more accessible to a wider audience.
- *URL validation:* Input validation was implemented, using the validator's Python package, to ensure only valid URLs are submitted for prediction.
- *Feedback loop:* After each prediction, the user was asked for feedback (thumbs up/down), which was logged and will be used in post-analysis.
- *Persistent logging:* All predictions were logged in a local SQLite3 database (phishing_logs.db) for timestamped logging of predictions, confidence score, user session, and user feedback. This provides an audit trail for future reference and provides some history for possible retraining.
- *Log viewing and management:* Logged data were displayed in the Streamlit app's interface in a table format and a "Clear Logs" button is included in the application to give users the option to clear (reset) stored logs if needed.
- *Batch and individual prediction interfaces:* The user interface supports real-time single URL prediction and bulk processing, with visual output and the option to download results.

- *Automation support:* A run_app.bat script was written in order to facilitate deployment and execution directly from the Windows desktop environment.

This fully implemented real-time deployment meets the objective of performance and is available for use (transparency). It supplied an intuitive and real-time feedback interface, is bilingual, and is interactive with visual explanations which adequately establishes an important scaffolding for the possibility of growing as a browser extension or publicly available web service in the future as a contribution to an output prediction (phishing or legitimate).

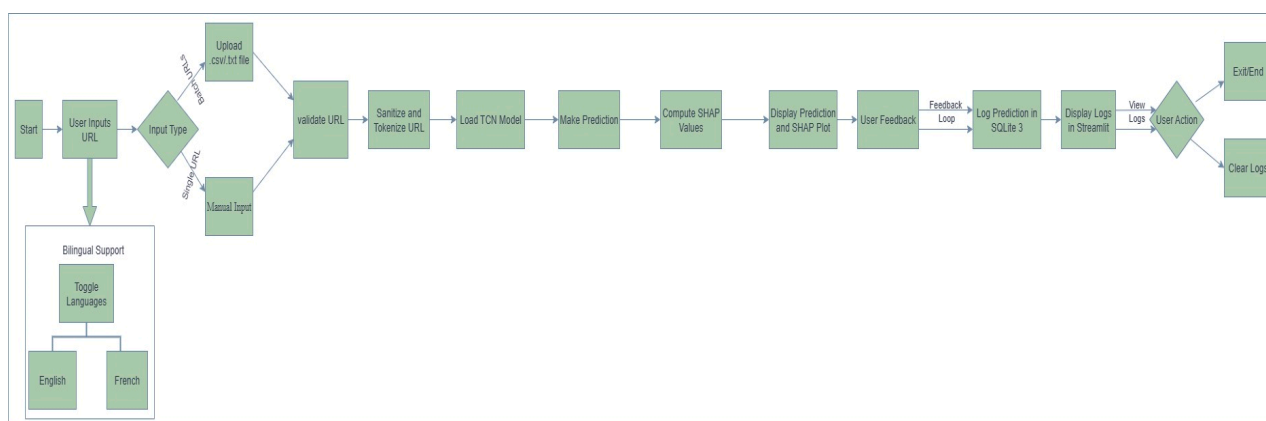


Figure 8. Architecture of the real-time deployment of the proposed model.

Algorithm 2: Phishing Detection with TCNWithAttention

To facilitate real-time classification in the Streamlit web application, we designed a phishing detection pipeline based on the proposed Temporal Convolutional Network with Attention (TCNWithAttention) architecture detailed in Section 5.4. As shown in Algorithm 2, the framework takes a sanitized URL, encodes it to a fixed length character sequence, applies the TCN backbone for temporal pattern learning, and integrates an attention pooling layer to accentuate the most suspicious parts of the URL. Finally, the model generates a phishing probability score, along with SHAP-based explanations and visualizations of attention weights for greater interpretability upon analyst review.

Algorithm 2: TCNWithAttention Phishing Detection (D, char2idx, MAX_LEN)

Inference Phase

1: For each new incoming URL (real-time or batch):

2: sanitized_url $\leftarrow$ Sanitize(input_url)

3: tokenized_seq $\leftarrow$ T.encode(sanitized_url)

4: X $\leftarrow$ Embed(tokenized_seq)

5: H $\leftarrow$ M.TCN(X)

6: c $\leftarrow$ M.Attention(H)

7: $\hat{y} \leftarrow$ Sigmoid(FC(c))

8: If $\hat{y} > 0.5$ then prediction $\leftarrow$ “Phishing” else prediction $\leftarrow$ “Legitimate”

9: shap_values $\leftarrow$ SHAP_Explain(M, tokenized_seq)

10: Display prediction, confidence score, and SHAP + Attention visualization

Return: prediction, $\hat{y}$, attention weights, shap_values

8. Conclusions

This paper addressed the increasing challenge of phishing by creating a real-time URL-only detection system for brand protection through a lightweight, interpretable, and scalable deep learning system. Originally, we intended to have a multi-modal solution with text and image (for example, logo detection, website content, and logos metadata), but after analyzing the most up-to-date literature (2024–2025), we discovered that a character-level neural model using only URLs was even better suited for detection purposes, especially for zero-day phishing in real-time, and it acted as a strong external dataset.

The final system was implemented with our custom model, the Temporal Convolutional Network with Attention (TCNWithAttention) structure that combined dilated residual convolutions with an attention pooling layer, so that the system could learn these more complex sequential patterns across the URLs and yet still be interpretable. The model was trained using our original GramBeddings dataset and tested using the PhiUSIIL (Phishing URL) external dataset with an F1-score of 97.16%, achieving high performance and outperforming other architectures such as CNN, BiLSTM, and BERT in terms of processing speed and accuracy.

This is a lightweight deployed solution, which was created to run on CPU, which would suit the majority of enterprise environments, or settings with limited processing resources. This solution is slightly unique, in that it provides real-time interface functionality, is web-based using streamlit, and is interpretable using SHAP explainability and attention heatmaps for transparency. This will give cybersecurity analysts the ability to inspect not only the prediction but the SHAP values and attention weights, highlighting the segments of the original URL (e.g., subdomain deception, brand identity deception attempts, suspicious tokens, etc.) that were the reasons for the prediction.

In addition to model performance, explainability was a primary focus of this study. The attention visualizations consistently corresponded with established phishing techniques, and the SHAP analysis further supported evidence of the model's reasoning. The research of false positives and false negatives also identified semantic, structural edge cases, which could be further improved, for example, with internationalized domain names and adversarial adaptation.

This project was able to successfully develop a real-time phishing detection system that balanced speed, accuracy, and explainability. The proposed TCNWithAttention model, with its character level and explainable approach, demonstrates that deep learning systems for URLs can provide pragmatic, scalable solutions in the space of brand protection within security ecosystems, when developed intentionally with explainability methods. This work is the first to unify TCNs, attention, and SHAP into a deployable URL-only phishing detection framework, bridging academic innovation with real-world, interpretable cybersecurity solutions.

8.1. Key Achievements

The work conveys the following key contributions:

- **Lightweight Architecture:** Created a new TCNWithAttention framework that employs character-level URL modeling for real-time phishing detection.
- **Benchmark Validation:** Established superior performance compared to CNN, BiLSTM, and BERT baselines, gaining a balance between performance, inference speed, and efficiency.
- **Explainability:** Provided dual-layer explainability, using attention visualizations and SHAP analysis, providing trust for analyst and compliance requirements.
- **Deployability:** Developed a CPU-only, Streamlit-based prototype that enables single URL and batch URL classification, bilingual support, and analyst feedback collection.

- **Robustness:** Established resilience—validated on an external dataset, PhiUSIIL, and effective against zero-day and obfuscated phishing URLs.

8.2. Future Work

Several future work avenues seem promising. The strongest area will continue to be adversarial robustness, as there are constant new attacks with evolving obfuscation mechanisms, including, but not limited to, homoglyphs, internationalized domain names (IDNs), and adversarial-formed substrings. Lab experiments with synthetic URLs and adversarial modified URLs will be essential for improving robustness and resilience. Another area to focus on is multilingual capability, which would increase detection capabilities in international contexts and non-Latin script detection.

Another important line of inquiry would be strengthening integration into ecosystem security practices. Though our model could serve as an excellent proactive first line of defense, the performance is only maximized when existing layers of security are also implemented. For example, the URL classifier could run first and could provide a pre-filtering step prior to MFA being initiated, while also alerting a security monitoring dashboard (SIEM) to correlate the URL with other signs of intrusion. Other behaviors could include the URL classifier augmenting existing multimodal systems by serving as a fast URL-level screening tool and reserving the more costly HTML or logo-level analysis for high-risk cases.

Researchers might consider a closed-loop learning process that would enable the classifier to incorporate some form of analyst or user input into retraining processes. This would be essential in evolving to adapt to organization zero-day phishing circumstances. Future work also includes exploring robust interpretability frameworks to provide an analyst with actionable insights and explanations, while preventing possible adversaries from discovering specific detection heuristics. This could include roles or partial access to explanation, differential privacy explanation algorithms, or adversarial robustness testing for interpretation to ensure interpretability tools do not dilute a model or make the system exploitable. All these considerations in future work provide robustness, resilience, and preparation in implementing URL-only detection to support enterprise-grade, multi-layered cybersecurity defensive strategies.

Author Contributions: M.-L.E.A.: writing—original draft, software, conceptualization, methodology. S.D.: writing—review and editing, validation, supervision and conceptualization. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lin, Y.; Liu, R.; Divakaran, D.M.; Ng, J.Y.; Chan, Q.Z.; Lu, Y.; Si, Y.; Zhang, F.; Dong, J.S. Phishpedia: A hybrid deep learning based approach to visually identify phishing webpages. In Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), Online, 11–13 August 2021; pp. 3793–3810.
2. Li, Y.; Huang, C.; Deng, S.; Lock, M.L.; Cao, T.; Oo, N.; Lim, H.W.; Hooi, B. KnowPhish: Large Language Models Meet Multimodal Knowledge Graphs for Enhancing Reference-Based Phishing Detection. In Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24), Philadelphia, PA, USA, 14–16 August 2024; pp. 793–810.
3. Opara, C.; Wei, B.; Chen, Y. HTMLPhish: Enabling phishing web page detection by applying deep learning techniques on HTML analysis. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8.

4. Bozkir, A.S.; Dalgic, F.C.; Aydos, M. GramBeddings: A new neural network for URL based identification of phishing web pages through n-gram embeddings. *Comput. Secur.* **2023**, *124*, 102964. [\[CrossRef\]](#)
5. Bai, S.; Kolter, J.Z.; Koltun, V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv* **2018**, arXiv:1803.01271. [\[CrossRef\]](#)
6. Le, S.; Markopoulou, A.; Faloutsos, M. URLNet: Learning a URL representation with deep learning for malicious URL detection. *arXiv* **2018**, arXiv:1802.03162. [\[CrossRef\]](#)
7. Xie, L.; Zhang, H.; Yang, H.; Hu, Z.; Cheng, X. A scalable phishing website detection model based on dual-branch TCN and mask attention. *Comput. Netw.* **2025**, *263*, 111230. [\[CrossRef\]](#)
8. Do, N.Q.; Selamat, A.; Krejcar, O.; Fujita, H. Detection of malicious URLs using Temporal Convolutional Network and Multi-Head Self-Attention Mechanism. *Appl. Soft Comput.* **2025**, *169*, 112540. [\[CrossRef\]](#)
9. Almujaheed, N.F.; Haq, M.A.; Alshehri, M. Comparative evaluation of machine learning algorithms for phishing site detection. *PeerJ Comput. Sci.* **2024**, *10*, e2131. [\[CrossRef\]](#) [\[PubMed\]](#)
10. Guo, W.; Wang, Q.; Yue, H.; Sun, H.; Hu, R.Q. Efficient phishing URL detection using graph-based ML and loopy belief propagation. *arXiv* **2025**, arXiv:2501.06912.
11. Mishra, R.; Varshney, G. A study of effectiveness of brand domain identification features for phishing detection in 2025. *arXiv* **2025**, arXiv:2503.06487. [\[CrossRef\]](#)
12. Zia, M.F.; Kalidass, S.H. Web Phishing Net (WPN): A scalable machine learning approach for real-time phishing campaign detection. In Proceedings of the 2024 4th Intelligent Cybersecurity Conference (ICSC), Valencia, Spain, 17–20 September 2024; pp. 206–213.
13. Zhou, J.; Zhang, K.; Bilal, A.; Zhou, Y.; Fan, Y.; Pan, W.; Xie, X.; Peng, Q. An integrated CSPPC and BiLSTM framework for malicious URL detection. *Sci. Rep.* **2025**, *15*, 6659. [\[CrossRef\]](#) [\[PubMed\]](#)
14. Baskota, S. Phishing URL Detection using Bi-LSTM. *arXiv* **2025**, arXiv:2504.21049. [\[CrossRef\]](#)
15. Mangalam, K.; Subba, B. PhishDetect: A BiLSTM based phishing URL detection framework using FastText embeddings. In Proceedings of the 2024 16th International Conference on COMMunication Systems & NETworks (COMSNETS), Bangalore, India, 3–7 January 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 637–641.
16. Chen, Y.H.; Chen, J.L.; Chiu, S.P. AI-Enhanced Phishing Detection: Leveraging Transformer-Based Representations with GAN for Robust Security. In Proceedings of the 2025 International Conference on Advanced Communications Technology (ICACT), Pyeong Chang, Republic of Korea, 16–19 February 2025; pp. 156–161.
17. Aljofey, A.; Bello, S.A.; Lu, J.; Xu, C. Comprehensive phishing detection: A multi-channel approach with variants TCN fusion leveraging URL and HTML features. *J. Netw. Comput. Appl.* **2025**, *238*, 104170. [\[CrossRef\]](#)
18. Cao, T.; Huang, C.; Li, Y.; Huilin, W.; He, A.; Oo, N.; Hooi, B. PhishAgent: A robust multimodal agent for phishing webpage detection. *Proc. AAAI Conf. Artif. Intell.* **2025**, *39*, 27869–27877. [\[CrossRef\]](#)
19. Lundberg, S.M.; Lee, S.I. A unified approach to interpreting model predictions. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 4768–4777. Available online: <https://arxiv.org/abs/1705.07874> (accessed on 11 June 2025).
20. Prasad, A.; Chandra, S. PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning. *Comput. Secur.* **2024**, *136*, 103545. [\[CrossRef\]](#)
21. Swamy, A.A.; Kumar, D. Phishing Detection Using a Convolutional Neural Network Model on Website URLs. In Proceedings of the 2024 International Conference on Sustainable Technology and Engineering (i-COSTE), Perth, Australia, 18–20 December 2024; pp. 1–8.
22. Calzarossa, M.C.; Giudici, P.; Zieni, R. Explainable machine learning for phishing feature detection. *Qual. Reliab. Eng. Int.* **2024**, *40*, 362–373. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.