technologies

MDPI

Article

MiniLM-CNN-LSTM: A Lightweight Hybrid Transformer Model for Malicious URL Detection

Emad-ul-Haq Qazi [1,2,*], Muhammad Hamza Faheem [1] and Abdulrazaq Almorjan [2]

[1] Department of Cybersecurity and Digital Forensics, College of Forensic & Investigative Sciences, Naif Arab University for Security Sciences, Riyadh 14812, Saudi Arabia
[2] Center of Artificial Intelligence for Security, Naif Arab University for Security Sciences (NAUSS), Riyadh 14812, Saudi Arabia
* Correspondence: qabdulrab@nauss.edu.sa

Abstract

Phishing and malicious websites are a serious threat on the internet. Attackers use fake links to trick users and steal their private information. Detecting these links is difficult because attackers change their tricks often. Many old methods cannot detect new or hidden threats. Some recent models use deep learning (DL), but they are large, slow, and hard to use in real-time systems. In this paper, we present a lightweight and accurate model called MiniLM-CNNLSTM. It combines a small transformer model (MiniLM) with a hybrid DL network using Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) layers. The transformer learns the meaning of URLs. The CNN finds important patterns. The LSTM captures the order of characters. We also add handcrafted features that help the model detect tricky URLs. We test our method on two public datasets: the Phishing Site URLs dataset and the Malicious URLs dataset from Kaggle. We use 3-fold cross-validation and early stopping to ensure fair and stable results. The MiniLM-CNN-LSTM model outperformed previous benchmarks by achieving an average three-fold cross-validation accuracy of 98.98%, a precision of 98.63%, a recall of 98.29%, an F1-score of 98.46%, and a false positive rate of 0.68%. The proposed model has a higher accuracy, precision, recall, F1-score and a lower false positive rate, which enhances the accuracy by 1.88, precision by 3.77, recall by 4.17 and decreases the false positive rate by 61.58% compared with the strongest baseline (Distil BERT + CNN-LSTM), showing significant practical improvements. The results show that our approach is fast, small, and highly effective. It can detect phishing and malicious links with high accuracy. This makes it a good choice for real-time security systems like browsers, email filters, or firewalls.

Keywords: malicious URL detection; phishing detection; lightweight transformer; MiniLM; Convolutional Neural Network (CNN); Long Short-Term Memory (LSTM); deep learning

Check for updates

Academic Editors: Sikha Bagui, Sotirios K. Goudos, Fazlullah Khan and Ateeq Ur Rehman

Received: 19 January 2026
Revised: 14 April 2026
Accepted: 3 May 2026
Published: 24 May 2026

Copyright: © 2026 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.

1. Introduction

The widespread use of the internet in our daily lives has led phishers to develop sophisticated new scam tactics. Ranging from banking and medical services to communications and even entertainment, people depend on the World Wide Web (WWW) to perform a myriad of tasks [1,2]. While having a digital presence has its advantages, it has also facilitated the development of various cybercrimes [3,4]. One of the most dangerous ones is the adoption of malicious hacking websites, known as phishing websites [5]. They are built with the intent of tricking individuals into giving sensitive details such as usernames,

passwords, banking details, and other private information. Phishing links masquerading as benign sites are sent through emails, messages, or advertisements and are meant to look official [6–8]. Thus, the majority of unsuspecting individuals who click on these links are bound to fall into the fraudulent trap, which is why phishing is regarded as one of the most popular and effective cyber-attacks [9,10]. The detection of malicious URLs is a daunting task because attackers are constantly changing their schemes. Attackers employ URL obfuscation, IP-based links, redirection, and even typosquatting to obscure their true motives [11,12]. Systems that solely rely on blacklisting URLs are obsolete, as such systems depend on databases of URLs that are known, which tend to lag evolving threats. Similarly, structural modifications to the URLs are enough to bypass rule-based filters. To solve this problem, a few researchers have suggested URL classification models with the assistance of machine learning, which include the presence of digits, special characters, or suspicious substrings [13–15]. However, those models are lacking in the ability to capture the deep semantic meaning of sequences and patterns in the URL.

New approaches to the problem have emerged recently thanks to advancements in artificial intelligence (AI). There is a growing optimism towards solving cybersecurity challenges with deep models like Convolutional Neural Networks (CNNs) [16], recurrent neural networks (RNNs) [17], and transformers [18]. Among these models, one of the most promising is the transformer model; it can analyze complex contextual relationships between strings of words, helping in the evaluation of URL strings. The drawbacks of such full-size models, however, preclude their practical use; for instance, BERT is too costly in terms of computational resources for real-time applications or low-power settings [19,20]. On the other hand, while CNNs and Long Short-Term Memory (LSTM) capture important structural and sequential information, critical, deeper semantic relationships are not captured [21,22]. Thus, a lightweight blend of these models will help achieve the optimal trade-off between performance and practicality. So far, the issue of designing a hybrid model that captures all these features efficiently has yet to be tackled.

Even with the advancements in URL-based threat detection, the accessibility of the existing models still poses concerns about their practicality for use outside the lab. Models such as BERT and RoBERTa possess large transformers that offer superior semantic understanding [23,24]. However, these models also need an enormous amount of memory and computational power, making them impractical for use in browser-based extensions or mobile security applications. Meanwhile, CNNs and LSTM models excel at learning local and sequential patterns, but they do not take into consideration the contextual meaning of the tokens [25]. In addition, the absence of more advanced methodologies and basic lexical characteristics that are easy to acquire but enhance detection is a major negligence [26]. The preference of other models is to exhibit poor generalization because of overfitting or a lack of reasonable evaluation methods [27]. These issues have led to the need for a model that disregards these limitations and employs lightweight logic, context recognition, pattern recognition, and tunability.

As a way of bridging these gaps, this paper suggests a new, lightweight hybrid deep learning (DL) model named MiniLM-CNN-LSTM to detect malicious URLs. The new model is based on the MiniLM transformer as the source of contextual embeddings that are generated using the raw URLs as the raw inputs. The local n-gram-like patterns are captured with the help of a CNN and followed by the LSTM layer with these embeddings. The LSTM considers the sequence of characters and symbols in the URL. Besides these deep features, we also introduce some handcrafted features like the number of dots, the presence of an IP address, and the length of the URL. The combination enables the model to identify the meaning and structure of the URLs, which enhances its performance. The architecture is small and quick and can perform its work almost in real time, even on low-end hardware.

Moreover, we also evaluate the performance of the model based on two publicly available datasets: the Phishing Site URLs dataset and the Malicious URLs dataset of Kaggle. These datasets contain URLs of different levels of risk and offer a feasible assessment of our approach. We perform 3-fold cross-validation by partitioning the data into different data partitions to ensure fairness in the model evaluation. In addition to this, we use early stopping to enhance generalization and prevent overfitting. Result presentation is performed with the help of accuracy, precision, recall, F1-score, and other conventional measuring benchmarks along with confusion matrices and ROC curves. MiniLM-CNN-LSTM is fast and precise, which is why it can be easily applied to the real world and is useful to cybersecurity devices such as browser extensions, email filters, or intrusion detectors. The design and analysis of the model involve the following:

1. An innovative hybrid DL model with a lightweight transformer (MiniLM) and CNN and LSTM layers to effectively and accurately detect malicious URLs.
2. A combination of lexical characteristics that are handcrafted with deep semantic embeddings to enhance the detection performance and interpretability.
3. Optimization of transformer layers and a bespoke architecture, which is lightweight and ready to deploy.
4. Detailed analysis based on two real-world datasets with 3-fold cross-validation, early stopping, and various measures such as accuracy, precision, recall, F1-score, confusion matrix, and ROC curve.
5. The model exhibited high performance, demonstrating the potential of this architecture for real-time security systems.

Despite the existence of prior hybrid transformer-based phishing detection models, the novelty of the proposed MiniLM-CNN-LSTM architecture lies at the system-design level rather than the isolated component level. Unlike prior works such as Zaimi et al. [5], which primarily relied on DistilBERT embeddings and hybrid CNN-LSTM classification, our approach differs in four fundamental aspects. First, instead of utilizing only the [CLS] token representation for classification, the complete contextual token sequence from MiniLM is preserved and processed, enabling richer structural and localized pattern learning. Second, the architecture is explicitly designed with deployment considerations, incorporating resume-capable cross-validation, early stopping mechanisms, memory-efficient embedding strategies, and lightweight inference objectives to facilitate real-time integration in constrained environments. Third, the model emphasizes the reduction of false positives rather than accuracy alone, recognizing that in practical cybersecurity systems, minimizing benign URL misclassification is critical for user trust and operational stability. Finally, the lightweight MiniLM backbone is systematically validated under low-resource training and inference conditions, reinforcing the model's real-world deployment intent rather than purely laboratory-level performance optimization. These system-level design choices distinguish the proposed work from prior hybrid transformer-based models.

In the remainder of this paper, Section 2 will conduct a review of recent articles on phishing URL detection with emphasis on transformer-based and hybrid deep learning. Section 3 describes the MiniLM-CNN-LSTM architecture, the proposed methodology, and the experimental setup, including the datasets used. Section 4 presents and discusses the performance results, compares the proposed model with the state-of-the-art baselines, and concludes the paper with directions for future work.

2. Related Work

Asiri et al. [28] have suggested a hybrid design, which parses the primary URL of a web page and the embedded URLs (in hyperlinks and in iframes) with a CNN model to extract the local features and with a transformer encoder to extract the long-range depen-

dencies. The model has a high F1-score, indicating its effectiveness in representing textual and structural patterns, but the URL extraction is exhaustive, which may increase computational cost and introduce latency when the model is applied in real time. Otieno et al. [29] describe how phishing URL contextual nuances are captured by fine-tuning a pretrained BERT model on URL data, avoiding large-scale feature engineering at the expense of being overtrained on large labeled training sets.

N. Q. Do et al. [3] enhance the deep learning classifiers with a pretrained transformer to project a unified framework that can lead to the achievement of high-quality processing of massive URL collections. The model has a strength in the fact that the combination of complementary classifiers enhances the performance of detection. However, increased model complexity due to added layers may slow down time-sensitive inference scenarios. A. B. Majgave and N. L. Gavanka [30] develop an innovative model that combines transformer-based features with a Deep Belief Network (DBN) to detect phishing websites, achieving improved detection rates compared to traditional heuristic approaches. Despite its strong performance, the complex integration of different deep learning modules may hinder its deployment in environments with limited computational resources. J. KS et al. [31] propose a system that processes phishing URLs in real time through a Chrome extension using a Knowledge-Distilled ELECTRA model, which offers nearly perfect accuracy. The perpetual learning loop with real-time alerts is innovative; however, how the system withstands the agility of persistent threats over time and the computational cost for periodic retraining to maintain system performance are valid concerns. H. Ghalechyan et al. [32] investigates both deterministic and probabilistic approaches to neural networks for URL classification, establishing a baseline of solid performance for evaluation by transformer-based methods. While this does reinforce the argument in favor of deep learning with its pattern recognition for URLs, its lack of examination of other transformer architectures is concerning from a scalability and contextual knowledge perspective.

A. B. Majgave and N. L. Gavanka [30] create an automatic phishing website detection model built on a transformer feature extractor and a Deep Belief Network (DBN), achieving higher detection rates than heuristic methods. Even with such strong performance, the elaborate composition of multiple layers of deep learning modules can be a barrier to systems with limited computational power because of their resource-intensive nature. J. KS et al. [31] use attention mechanisms along with GRU networks to capture both the sequential and contextual dynamics of URLs and detail the important aspects of attention mechanism benefits in phishing pattern detection. As mentioned in this work, the attention mechanisms are beneficial; however, attention-augmented GRU models, in general, have fewer parallel processing capabilities when compared to transformer architectures, which makes scaling to larger datasets less efficient. S. Jamal et al. [33] obtained URL detection insights by applying off-the-shelf transformer models DistilBERT and RoBERTa for email classification, achieving high precision, recall, and F1-scores. Despite the strong results achieved, focusing on email data creates a need for further adjustments to generalize the results to scenarios based on URLs.

S. K. Birthriya et al. [34] propose a hybrid model that combines Gated Recurrent Units (GRUs) with Convolutional Neural Networks (CNN). The model leverages the temporal feature extraction capabilities of GRUs together with the spatial feature extraction strengths of CNNs to achieve 99.0% accuracy on a Kaggle dataset of over 2.5 million URLs. The authors also discuss the model's potential for real-time deployment and its advantages over standalone GRU or CNN models in capturing complex URL patterns. W. Guo et al. [35] proposed a graph-based model that adds network-level features like IP addresses and authoritative and name servers to the typical URL structure features. The approach

employs Loopy Belief Propagation with an improved convergence strategy to capture dependencies in the URL graph effectively. The findings show a high F1-score, demonstrating the model's efficacy in adversarial conditions. Transform-based architectures have been adopted widely for more accurate detection of phishing URLs. P. H. Hussan and S. M. Mangj [36] employed a teacher–student paradigm of DistilRoBERTa and RoBERTa for phishing URL identification, which increased the model's computational efficiency. However, the model's strength against adversarial URL obfuscation has yet to be evaluated.

A. Aljofey et al. [37] proposed BERT-PhishFinder, combining optimized DistilBERT and fine-tuned hyperparameters. The proposed study outperforms traditional approaches in URL-based phishing detection. Its drawback is the reliance on labeled datasets, which limits generalizability. P. Chaudhary et al. [38] studied phishing detection through Rancorous URL Classification's annotation-based interpretability, comparing multiple transformer models for phishing detection. Their dataset was static over time, which limits the reliability of conclusions drawn about evolving threats. R. Zaimi et al. [5] developed a hybrid phishing URL detection model using DistilBERT for text feature extraction with a CNNLSTM for classification. The model experimented on two large publicly available datasets under two experimental conditions, one with word embeddings alone, and the other with embeddings and 21 URL features. The second case had a more precise result that proved the strength of the combination of semantic and structural information used in URLs beyond the basic features. Despite the good performance of the model, the high computation cost restricts its usage in real time, including the possibility of classifying some of the complex structured legitimate URLs with long directory structures or multiple subdomains as phishing links.

Although there has been significant advancement in phishing URL detection using transformers, as noted in the comparative evaluation in Table 1, there are still critical gaps in the identified solutions. Many existing studies either focus exclusively on identifying textual patterns using transformer encoders or on bespoke lexical feature extraction, hardly ever employing both methodologies in tandem. Contextual embedding models such as BERT and RoBERTa are available; however, they pose significant computational constraints, rendering real-time use impractical. Additionally, some models utilize hybrid CNN GRU frameworks to extract local and sequential URL features but fail to utilize semantic representations. Certain high-performing systems are constrained by large labeled datasets, becoming increasingly complex to adapt to obfuscated adversarial tactics and evolving attack methodologies. Therefore, there is a need to develop an expressive model that encapsulates semantic comprehension, sequential feature recognition, and handcrafted statistical elements, thus informing the construction of our proposed architecture, MiniLM-CNN-LSTM.

Table 1. Comparison of transformer-based approaches for phishing URL detection, highlighting architectural design and deployment limitations.

Ref	Model Type	Highlights	Limitations
[28]	CNN + Transformer	Analyzes main and embedded URLs, combining local and long-range pattern extraction	High computational overhead; potential latency
[29]	Fine-tuned BERT	Deep contextual embeddings eliminate manual features and capture syntax and semantics	Dependent on large, high-quality labeled data; adversarial vulnerability
[3]	Transformer + DL Classifiers	Combines classifiers to boost detection on large datasets	Increased model complexity; slower inference

Table 1. *Cont.*

Ref	Model Type	Highlights	Limitations
[39]	Transformer + Explainability	LIME and Transformer Interpret aid interpretability in phishing detection	Transfer to URL domain requires domain-specific features
[40]	ELECTRA + Chrome Extension	Real-time alerts and continuous learning with high accuracy	Vulnerable to evolving threats; retraining overhead
[32]	DL (DNN/PNN)	Provides baselines against which transformer models are compared	Does not explore transformer-based scalability
[30]	Transformer + Deep Belief Network	Higher detection accuracy than heuristics via hybrid modeling	Integration complexity limits low-resource deployment
[31]	GRU + Attention	Captures sequential and contextual features in URLs	GRUs less efficient than transformers on large datasets
[33]	DistilBERT/RoBERTa (Email)	High precision and recall in phishing email classification	Needs adaptation for URL data
[34]	GRU + CNN	Achieves 99% accuracy on large Kaggle dataset: real-time potential	No transformer benchmark comparison
[35]	Graph-Based (Loopy Belief)	Integrates network and structural features; strong F1 in adversarial cases	Complex graph processing may hinder deployment
[36]	DistilRoBERTa + RoBERTa	Efficient teacher–student design improves learning	Lacks evaluation against obfuscated URLs
[37]	DistilBERT (Optimized)	Outperforms legacy models with fine-tuned hyperparameters	Generalization is limited by dataset dependency
[38]	Multi-model Transformers	Comparative analysis emphasizing interpretability	Lacks dataset temporal diversity
[5]	DistilBERT + CNN-LSTM	Uses word embeddings + 21 features, achieves 98.19% accuracy	False positives in complex legitimate URLs; runtime constraints

3. Proposed Methodology

Using a low-power hybrid technique, a model designed for malicious URL detection, referred to as MiniLM-CNN-LSTM, is proposed to identify harmful URLs. The pipeline begins with a raw URL which is first tokenized and then sent to MiniLM, which does the embedding. The embeddings are subjected to CNN and LSTM, where patterns and sequences are learned. In parallel, handcrafted features are extracted from the raw URL. Finally, all features are fused and passed through fully connected layers to classify the URL as malicious or benign. The motivation of this hybrid architecture is that the three components complement one another: MiniLM retrieves global semantic data, CNNs capture localized structural patterns, and LSTMs capture sequential dependencies, which renders the architecture very suitable to complex and obfuscated URL structures. It should be noted that to provide a fair assessment, all the baseline models were set up in the same environment, with the same data, preprocessing, tokenization length, training policy, and cross-validation folds.

An abstract overview of the model pipeline is shown in Algorithm 1. Each input URL undergoes tokenization via a transformer, extracting token IDs and an attention mask. Simultaneously, a set of handcrafted lexical features is constructed. The prepared input undergoes semantic embedding through the MiniLM encoder, which is further tokenized. The resultant embeddings are input into a CNN, where structural and sequential features are extracted through max-pooling followed by LSTM. The LSTM output, together with handcrafted features, forms a joint feature representation after concatenation. A dense layer with sigmoid activation receives this representation to compute the probability score. The model employs binary cross-entropy loss during training across all samples.

Algorithm 1. MiniLM-CNN-LSTM: URL classification workflow

Input: Dataset $\mathcal{D}$ = {(U_i, y_i)}_(i = 1)^N
Output: Prediction ^yi for each U_i
1 **for each** $(U_i, y_i) \in \mathcal{D}$ **do**
2 X_i, $M_i \leftarrow$ Tokenizer(U_i);
3 $F_i \leftarrow$ ExtractHandcraftedFeatures(U_i);
4 $E_i \leftarrow$ MiniLM(X_i, Mi);
5 $C_i \leftarrow$ CNN(E_i);
6 $P_i \leftarrow$ MaxPooling(C_i);
7 $H_i \leftarrow$ LSTM(P_i);
8 $Z_i \leftarrow$ Concatenate(H_i, F_i);
9 $\hat{y}_i \leftarrow \sigma$(Dense($Z_i$));
10 Train model by minimizing:
$\mathcal{L} = 1/N \sum_(i = 1)^N BCE(\hat{y}_i, y_i)$;

3.1. Overview of the Architecture

The proposed architecture, MiniLM-CNN-LSTM, is a hybrid model intended for the detection of malicious URLs that attempt to exploit both semantic and structural information, as shown in Figure 1. It has three major components: a transformer encoder (MiniLM), a hybrid CNN-LSTM network, and a tailored feature-fusion layer. Let us denote the input URL as U. The model begins with subword tokenization of U using a MiniLM tokenizer. The tokens are first fed into a pretrained MiniLM transformer encoder to obtain a contextualized embedding matrix, which is expressed as $E \in \mathbb{R}\{T \times d\}$, where T is the upper bound on the token length and d represents the embedding dimension:

$$E = MiniLM(U) \tag{1}$$

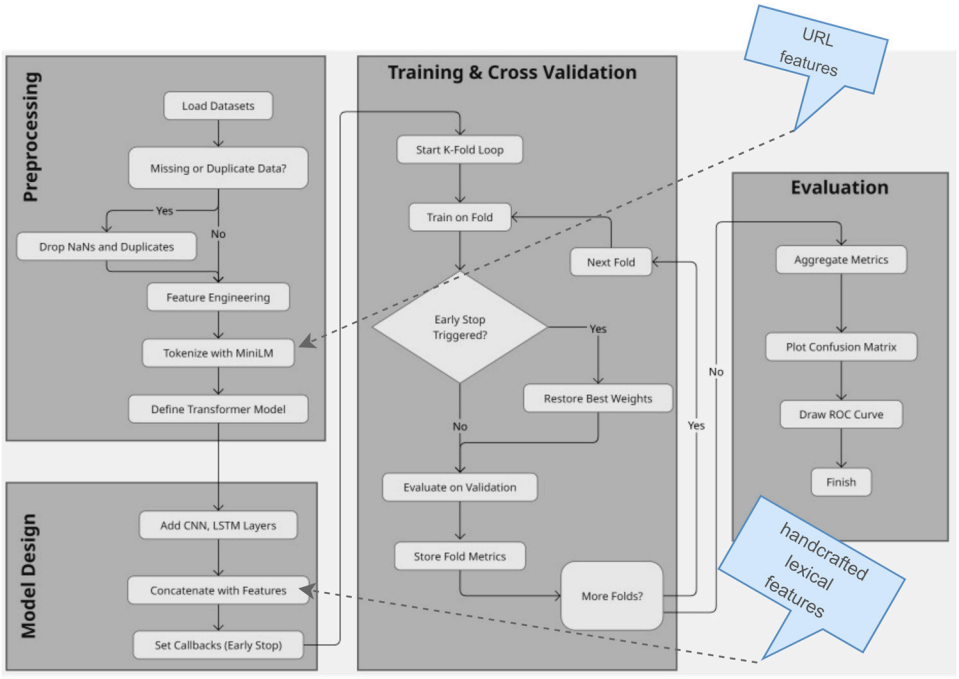


Figure 1. Proposed model architecture.

A lightweight transformer is employed in Equation (1) to capture semantic relations between the tokens. The output embeddings, E, are sent to a 1D CNN model that is used to extract local patterns from time-series data. A convolutional filter of size k is applied along the sequence dimension, leading to Equation (2).

$$C_i = ReLU(W_c * E_{i:i+k-1} + b_c) \quad (2)$$

where W_c and b_c are the filter weights and biases, respectively. The convolution operation is represented with the symbol $*$. This operation yields feature maps C, which capture substring-level structures of interest, such as suspicious patterns or keyword fragments. For learning the temporal dependencies, the output of the CNN is sent to an LSTM network:

$$H = LSTM(C) \quad (3)$$

With Equation (3), we model the sequence of the extracted patterns showing token interactions over time. The output H represents a fixed-length hidden summary of the sequence C. Simultaneously, a set of n handcrafted features are obtained directly from the URL U. These features are the length and the number of dots in the URL, as well as its lexical properties like the use of IP addresses. Let $F \in \mathbb{R}n$ denote the vector of handcrafted features. The deep LSTM output H is concatenated with F to create a unified representation, as shown in Equation (4).

$$Z = Concat(H, F) \quad (4)$$

The composite vector Z is subjected to fully connected layers, which yield the final prediction. The probability ^y that the URL is malicious is given using a sigmoid activation function $\sigma(\cdot)$ as:

$$(_\hat{y}([\![\ \hat{}\ ]\!]) = \sigma(Wz + bz) \quad (5)$$

In Equation (5), the model parameters for the layer output are W_z and b_z. The value given is ^y $\in$ [0, 1], which represents the probability of the input URL being malicious. The whole structure is designed to be light and efficient while ensuring accuracy. The model incorporates local pattern recognition and handcrafted features that combine with transformer-based contextual understanding.

Unlike transformer-only baselines, which rely solely on self-attention for sequence modeling, the proposed architecture explicitly separates global contextual modeling (MiniLM) from localized structural pattern extraction (CNN) and sequential refinement (LSTM). This layered design allows complementary feature learning across representation levels, which is particularly beneficial in URLs where local substrings (e.g., brand impersonation fragments, obfuscation tokens) interact with broader structural dependencies.

3.2. Data Representation and Preprocessing

In this part, we explain how raw URLs are modified and structured to serve as inputs for the proposed hybrid model. For processing of each URL, two parallel representations are created: a tokenized sequence for the transformer encoder and a handcrafted feature vector for lexical analysis. Denote U as a raw URL string. The initial process is tokenization, which is performed by the MiniLM tokenizer. It transforms U into a sequence of subword tokens $T = [t_1, t_2, \ldots, t_T]$, where T marks the upper or maximum sequence length. Each token is replaced with an Integer ID according to the translations in the MiniLM vocabulary. The tokenized sequence is then padded or truncated to a uniform fixed length, L, to ensure consistency across all inputs. The output of the process is a token-ID sequence, denoted by $X \in \mathbb{N}L$. An attention mask, $M \in \{0, 1\}\gamma L$, is also created to distinguish between

real tokens and padding along with token IDs. Both X and M are passed to the MiniLM transformer so that contextual embeddings can be generated as described in Equation (1).

In parallel, we also obtain a handcrafted feature vector $F \in \mathbb{R}n$ from the raw URL U. These features are crafted to capture shallow lexical patterns that are frequently associated with malicious URLs. The features integrated include URL length, count of dots (.), slashes (/), at symbols (@), equal signs (=), presence of an IP address, and count of query parameters. All numerical features are scaled by min–max normalization so that they lie within the interval [0, 1]. Doing so improves the model's learning capability and prevents instability due to variations in numerical values during training. The two parallel representations, (X, M) and F, are then provided in the transformer and parts of the architecture. Together, these representations form the full input for the MiniLM-CNN-LSTM model.

3.3. MiniLM Embedding Layer

The first component of the proposed model is a transformer encoder built on top of MiniLM. MiniLM is a distilled implementation of BERT that maintains most of the semantic understanding and is computationally less expensive. Intense technological operations where high speed and memory efficiency are required are its forte, which is highly relevant for real-time cybersecurity. For a tokenized URL sequence $X \in \mathbb{N}L$, a corresponding attention mask M $\in \{0, 1\}L$ is also provided. The MiniLM encoder produces contextualized token embeddings. The result is a dense embedding matrix $E \in \mathbb{R}L \times d$, where d represents the embedding dimension of MiniLM. The embedding is performed as shown in Equation (6):

$$E = MiniLM(X, M) \tag{6}$$

E is filled as follows: each row of E equals a token of the input sequence, containing both the value of the token and the meaning of the relationships with the neighboring tokens. Because of these properties, the model can identify brand-impersonating domains and other heuristics like constituent domains, keywords, and phrases. In comparison to other NLP tasks where sentence-level semantics are more important, URL detection is greatly aided by subtle character-level shifts, and MiniLM's attention mechanism is incredibly sophisticated and allows it to perform well even while using a domain-specific lexicon with exceptionally low deviating values. This enables a clear understanding of URLs disguised in complex forms or cunning coordinates of malevolent intent. The entire embedding matrix E is forwarded to the next step of the model, which has convolutional and sequential processing layers. Different from BERT-based methods, which frequently only utilize the [CLS] token for categorization, we utilize the entire sequence output because it contains rich details, both structural and localized, of the URL.

3.4. CNN Layer

The proposed model's CNN layer extracts local features from context embeddings generated by MiniLM. While transformer layers model global interactions among the tokens, CNNs excel at capturing the spatial structure or specific substrings and symbol combinations that occur in malicious URLs. For the output embedding matrix from the MiniLM encoder $E \in \mathbb{R}Lxd$, we use a 1D convolutional layer with m filters of size k. The embedding sequence is slid for each filter, and a new specific portrayal is created for each local window with the embedding window, as represented in Equation (7).

$$C_i = ReLU(W_c \times E_{i:i+k-1} + b_c) \tag{7}$$

The convolution operation applies filter weights Wc over the embedding matrix E, generating feature maps C that capture local structural patterns. Altering the windows E

leads to different feature maps C. C is, in turn, computed throughout the entire sequence of tokens, and C is a feature map that enhances specific portions of interest within the token embeddings. In this scenario, it shows the excerpts of embedding that are most important within the whole embedding and aid in token recognition.

Detecting such motifs is a strong suit of CNNs. Moreover, to enhance efficiency, we reduce feature-map dimensionality by applying a max-pooling operation over time. The max-pooling layer features a selection mechanism that captures the most salient activation of each feature map, ensuring the preservation of the most potent local signal:

$$P = MaxPool(C) \tag{8}$$

In Equation (8), the pooled output P retains the most salient activations from each feature map, preserving dominant local patterns while reducing dimensionality. The next step involves subjecting these outputs to an LSTM layer designed to capture relationships among these local patterns. Therefore, a feature extractor operates at the character level to lessen the sequence length fed into the LSTM, enhancing the hybrid architecture's performance while increasing model accuracy.

3.5. LSTM Layer

The output after max-pooling from the CNN layer is fed into an LSTM network. The function of the LSTM is to remember and hold temporal order information regarding the sequential dependencies within the localized structures captured by the CNN. With respect to the URL context, the sequence of the tokens and groups of characters is critical. For instance, a benign keyword within a suspicious context might suggest malicious intent. The model needs to be able to forget and remember information about these patterns in a temporal manner. Let us denote the pooled feature sequence from CNN as $P = [p_1, p_2, \ldots, p_m]$, where m is the length of the sequence post-pooling, $p_i \in \mathbb{R}f$ is the local feature vector, and m is the length of the sequence after pooling. The LSTM takes in a sequence at a time and updates its hidden and cell states, h_t and c_t, respectively, at each point in time. The computations in an LSTM, at time t, are done according to Equation (9).

$$\begin{gathered} f_t = \sigma(W_f p_t + U_f h_{t-1} + b_f) \\ i_t = \sigma(W_i p_t + U_i h_{t-1} + b_i) \\ o_t = \sigma(W_o p_t + U_o h_{t-1} + b_o) \\ \tilde{c}_t = tanh(W_c p_t + U_c h_{t-1} + b_c) \\ c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\ h_t = o_t \odot tanh(c_t) \end{gathered} \tag{9}$$

In Equation (9), ft, it, and ot denote, respectively, the forget, input, and output gates. Furthermore, $\tilde{c}t$ and ct are the candidate cell state and updated cell state, respectively, and h_t is the updated hidden state. Each gate has an associated weight matrix and bias given by $W\cdot$, $U\cdot$, and $b\cdot$, and elementwise multiplication is represented by $\odot$. The last hidden state hm is taken after processing the entire sequence and will serve as a compressed form of the whole input URL's structural pattern. This output, $H = hm$, holds information from local patterns established using CNNs as well as from sequential dependencies captured by the LSTM.

3.6. Handcrafted Feature Fusion

We also integrate a set of handcrafted lexical features based on the URL string U alongside the deep features obtained using the transformer-CNN-LSTM architecture. These features are determined using heuristic methods from previous work as well as practical cy-

bersecurity measures that suggest specific lexical patterns can indicate nefarious activity. Let $F = [f_1, f_2, \ldots, f_n] \in \mathbb{R}n$ denote the handcrafted feature vector for a given URL. Each element fi corresponds to a specific lexical characteristic and is normalized. The following features are included in our model:

- URL Length: Total count of characters in the URL.
- Number of Dots: Subdomain depth indicator, commonly exploited in phishing attacks.
- Presence of IP Address: Indicates whether part of the URL is an IP address instead of a domain name.
- Number of Slashes: Abnormally high slash counts may indicate hidden directories or obfuscation.
- Number of @ Symbols: Attackers sometimes redirect or use '@' to trick parsers.
- Number of Equal Signs: Indicates parameters that suggest embedded scripts or payloads.
- Number of Query Parameters: Phishing or tracking posed by complex query strings.

All features are scaled using min–max normalization to fall within the range [0, 1]. This guarantees coherence and mathematical reliability when integrated with deep features. After the handcrafted feature vector F is computed, it is concatenated with the deep LSTM output $H \in \mathbb{R}(d`)$ to produce a final joint representation Z.

$$Z = \text{Concat}(H, F) \tag{10}$$

The fused vector $Z \in \mathbb{R}(d` + n)$ represents high-level information of the deep model and human-understandable labels, blending high-level information with linguistically interpretable labels (see Equation (10)).

3.7. Fully Connected Layers and Output

For the proposed MiniLM-CNN-LSTM, the last connecting step includes one or more dense layers followed by a binary output unit. In this stage, the heterogenous features from the LSTM and the handcrafted features are fused into a vector $Z \in \mathbb{R}(d` + n)$ and transformed into a scalar ^y, which indicates whether the input URL is malicious or not. To enhance the expressiveness of the model, the first dense layer employs ReLU activation using Equation (11).

$$h = ReLU(W_1 Z + b_1) \tag{11}$$

Here, $W1 \in \mathbb{R}$^ (h x (d`+ n)) and $b1 \in \mathbb{R}$^h represent the weights and biases of the first dense layer and h is the hidden layer size. ReLU activation increases sparsity, which helps the model learn faster. To prevent overfitting and improve generalization, we drop this layer with dropout after the first dense layer. During the training phase, a percentage of the neurons are randomly turned off:

$$h = Dropout(h, p) \tag{12}$$

In Equation (12), p is the dropout rate, which is generally set around 0.3 and 0.5. This step in regularization prevents the model from relying on particular nodes in the network too heavily.

Finally, the second dense layer with a sigmoid activation function receives the output from the dropout layer. This layer transforms the intermediate vector into a probability value ^y $\in$ [0, 1], which determines how likely the input URL is malicious, as shown in Equation (13).

$$\hat{y} = \sigma(W_2 \tilde{} h + b_2) \tag{13}$$

where $W2 \in \mathbb{R}(1 \times h)$ and $b2 \in \mathbb{R}$ denote the weights and bias of the output layer, respectively, and $\sigma(^{\circ}\S)$ is the sigmoid function. The model attempts to minimize the binary cross-entropy loss corresponding to the predicted output $\hat{y}$ and the true output label $y \in \{0, 1\}$. The last prediction can either be set to 0.5, which is the threshold for binary classification, or adjusted according to the requirements of the application. This stage is fully connected and combines all crafted and learned features as a single output, thus completing the learning process of the MiniLM-CNN-LSTM architecture.

3.8. Training Strategy

Supervised learning is used to train the MiniLM-CNN-LSTM model on a labeled dataset containing URLs. Each URL has a binary label $y \in \{0, 1\}$ wherein 0 is assigned to a benign URL, while 1 is assigned to a malicious one. The objective is to minimize the binary cross-entropy loss, where the predicted probability is ^y and the true label is y.

3.8.1. Loss Function and Optimization

The approach selected for the loss function is binary cross-entropy, which is presented in Equation (14).

$$\mathcal{L} = -[y^{\circ}\S\log(\hat{y}) + (1 - y)^{\circ}\S\log(1-\hat{y})] \quad (14)$$

With this loss, more weight is assigned to the incorrectly predicted probability as it moves away from the true label. The parameters of the presented model are optimized with respect to the loss using the Adam W Optimizer, which is a version of Adam that incorporates weight decay regularization. It allows for the mitigation of overfitting without impacting the speed of convergence. The rate of learning is 1×10^{-5}, which is relatively low and enables the controlled tuning of pretrained transformer layers without the risk of forgetting entire learned features. All the layers of the MiniLM encoder are set as trainable to allow the model to adjust to the distribution of malicious and benign URLs.

3.8.2. Early Stopping and Regularization

The workflow implements early stopping to prevent overfitting and improve generalization. The model validation loss is checked for improvement after every epoch, and for a set number of epochs (patience = 2), no improvement must occur. In such case, the training process is stopped, and the model reverts to its best set of weights. Also, dropout regularization is implemented in the fully connected layers as specified in Equation (12) with a dropout of 0.5. It reduces co-adaptation between neurons and makes the model more robust to noise in the input.

3.8.3. K-Fold Cross-Validation

To mitigate potential bias arising from a single train–test split, we apply k-fold cross-validation with k = 3 to the dataset. The 3-fold cross-validation is chosen in order to balance the effectiveness of the computational process with the estimation of the performance, particularly considering the training cost of hybrid transformer-based architectures. The dataset is split into three equal sections. In each fold, two sections are utilized to train the model, while the third is used for validation. With this approach, the model undergoes several folds or iterations, drawing upon each fold's outcomes for improved averaging of results to raise overall performance metrics throughout the folds used incrementally. Three-fold cross-validation is chosen to balance efficiency with the model's generalization ability. Considering the architecture of the hybrid model consisting of a MiniLM transformer, CNN, LSTM, and handcrafted features, the time and resources needed to train each fold are significantly high. Additionally, 3-fold cross-validation enhances the evaluation process by optimizing the workload with sufficient variance to avoid monotony. The

model can be trained and validated on diverse subsets, which still capture the essence of the whole dataset. It is advantageous because the datasets are balanced and large. All the folds have enough samples to be used in all the training and testing stages. All three folds demonstrated consistent accuracy, precision, recall, F1-score, and ROC-AUC, confirming that the set number of folds provided unbiased performance metrics.

Data Leakage

The cross-validation was performed at the URL level using stratified sampling. While this setup reflects realistic URL filtering scenarios encountered in practical systems, it does not enforce domain-level separation across folds. Therefore, the possibility that URLs from the same domain appear in different folds cannot be fully ruled out. This limitation is acknowledged, and future work will investigate domain-aware and temporal splitting strategies to further strengthen generalization guarantees.

3.8.4. Batch Size and Epochs

A batch size of 128 is used during the training to get the gradient stable and still have a high level of efficiency in memory usage. The training is done with a maximum of 20 epochs, although early stopping will normally halt the training further, depending upon the validation performance. This systematic paradigm reflects the way the MiniLM-CNN-LSTM model should be trained so that it can achieve maximum accuracy and efficiency. Moreover, the reported results would never be biased or specific to a single data partition and, therefore, result in a higher credibility of the results.

3.9. Experimentation Setup

All experiments were done using Google Colab Pro, a cloud platform providing a GPU for accelerated computations. For this project, we utilized an NVIDIA L4 GPU (24 GB VRAM) along with a 12th Gen Intel Core i9 CPU@3.10GHz, backed by 52 GB of RAM. The development environment was based on Python 3.13 and was hosted on Ubuntu 20.04.6 LTS (Colab VM kernel). Working with neural networks was done with the frameworks TensorFlow 2.15 and Keras for building, training, and evaluation. The HuggingFace Transformers (v4.38.1) library was utilized to load and fine-tune the MiniLM-L12-H384-uncased model along with its tokenizer. For data handling, preprocessing, and feature manipulation, the libraries NumPy (v1.24) and Pandas (v1.5.3) were used with early stopping aimed at restoring the best weights to mitigate overfitting while using a batch size of 128. The model was set to use the Adam W optimizer, which is known for improving convergence while fine-tuning transformers due to its ability to decouple weight decay from gradient updates. A learning rate of 1×10^{-5} was maintained. Having all experimentation and coding run on a single Colab notebook automated reproducibility and enhanced system efficiency for rapid experimentation. Further details can be seen in Table 2.

Table 2. Experimental setup and hyperparameters for MiniLM-CNN-LSTM under 3-fold stratified cross-validation.

Component	Specification
Environment	Google Colab Pro
GPU	NVIDIA L4 (24 GB)
CPU	12th Gen Intel Core i9 @ 3.10 GHz
RAM	52 GB
OS Kernel	Ubuntu 20.04.6 LTS (Colab VM)

Table 2. *Cont.*

Component	Specification
Python Version	3.13
TensorFlow Version	2.15
Transformers Library	HuggingFace v4.38.1
Transformer Model	MiniLM (L12-H384-uncased)
Tokenizer	HuggingFace AutoTokenizer
Trainable	Transformer Layers Yes (All Unfrozen)
CNN Filters	256
Kernel Size	3
Max-Pooling	Pool Size = 2
LSTM Units	128
Dense Layer Units	128
Dropout Rate	0.5
Loss Function	Binary Cross-Entropy
Optimizer	AdamW
Learning Rate	1×10^{-5}
Batch Size 128	128
Max Epochs 20	20
Early Stopping Patience	2
Cross-Validation	3-Fold Stratified
Token Length	32
Feature Min–Max Scaling	Normalization
Output Activation	Sigmoid

3.10. Dataset Description and Transparency

Two publicly available benchmark datasets, namely the Phishing Site URLs dataset and the Malicious URLs dataset on Kaggle, were used to complete the experimental analysis. These datasets consist of labeled URLs (benign or malicious) gathered on various web sources. Both datasets were harmonized before training, so that there was consistency in labels. In particular, all the labels were converted to a single binary scheme, with benign URLs being assigned label 0 and the malicious ones being labeled 1. The two datasets were then combined into one corpus. Duplicate URLs were eliminated, and records that lacked a label were dropped to eliminate the possibility of performance inflation as a result of repeated samples. The overall combined collection of URLs also had 641,125 unique URLs after preprocessing and duplicate removal. Out of them, 428,080 URLs (66.77%) were classified as benign and 213,045 URLs (33.23%) were classified as malicious, which showed a medium class imbalance with benign samples. All the splits of data used stratified sampling to maintain this distribution of classes on the training and validation sets.

The MiniLM tokenizer was used to do tokenization with a fixed maximum sequence length of 32 tokens. Besides contextual embeddings, lexical representations were produced manually out of raw URLs, and structural characteristics (length of URL, number of subdomains, existence of IP addresses, number of special characters, etc.) were obtained. These numerical characteristics were normalized before the model was trained. Cross-validation of the dataset was done at the URL level with stratified cross-validation. Although the configuration is realistic in terms of URL filters, domain-level isolation was not implemented and is mentioned as a weakness in Section 5.

3.11. Input Representation and Tokenization

We tokenize URLs using the Microsoft/MiniLM-L12-H384-uncased tokenizer with a fixed maximum sequence length of 32 tokens. URLs longer than this limit are truncated,

while shorter sequences are padded. Although approximately 24.56% of URLs exceed 32 subword tokens (maximum observed length: 1185), truncation was adopted to ensure computational efficiency, memory stability, and consistent comparison across baseline models. Empirical inspection of token-length distributions indicated that discriminative lexical patterns are typically concentrated in domain prefixes, subdomains, and early path segments. Therefore, restricting sequences to 32 tokens preserves the most informative structural components while preventing excessive padding and training instability. This design choice aligns with prior transformer-based URL classification studies that prioritize prefix-dominant features.

4. Results and Analysis

The evaluation that was conducted in this section of the paper was based on the MiniLM-CNN-LSTM model using two benchmark datasets and three-fold cross-validation. Performance is benchmarked by the use of performance metrics, such as accuracy, precision, recall, F1-score, and false positive rate. To get more insight than the relevant training behavior, model consistency and classification ability, visual representations like ROC curves and loss/accuracy graphs and fold-wise utilization of measures are examined, which offer attractive insights into the performance of the model. The results are also compared with the results reported in the base paper [5] to bring out the improvements achieved. In order to test the performance of the proposed model, we compare it with the strongest baseline given the same experimental conditions. The proposed model surpasses the most powerful baseline (DistilBERT + CNN-LSTM) on all assessment parameters (see Table 3), especially on false positive rate reduction. All the models were tested with the same experimental conditions and the results have been reported as the average performance over three folds to guarantee reliability.

Table 3. Performance comparison under identical experimental conditions.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	FPR (%)
MiniLM-CNN-LSTM (Proposed)	**98.98**	**98.63**	**98.29**	**98.46**	**0.68**
DistilBERT + CNN-LSTM	97.10	94.86	94.12	94.29	1.77

4.1. Fold Training vs. Validation Accuracy

The training and validation accuracy of each of the three cross-validation folds are shown in Figure 2. The model has a clear upward moving trend in accuracy as it trains but the validation curves follow the training metrics closely, which implies that there is no overfitting. Each fold achieves an accuracy of over 99% after 10 epochs, and this shows the aggressive learned behavior of the model. Such restrictive constraints of training and validation accuracy indicate strong architecture performance across all folds.

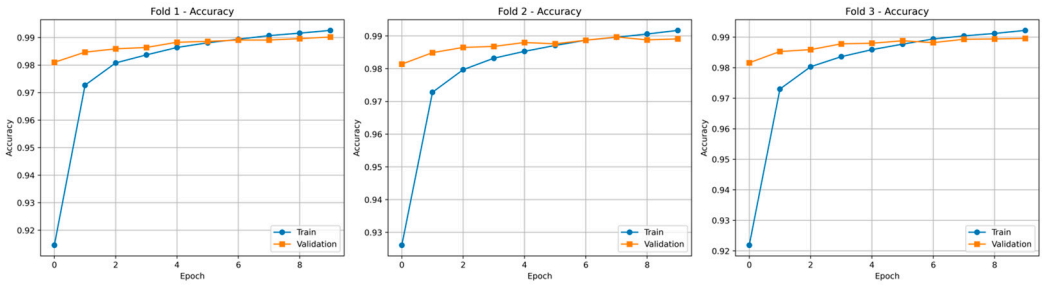


Figure 2. Training and validation accuracy curves for each fold.

4.2. Fold Training vs. Validation Loss

Figure 3 shows the loss of the training and validation across folds. The loss is in each instance non-increasing across epochs and validation loss is nearly identical to training loss. This steady reduction shows that the model is most likely learning the data and not overfitting the training data. It is worth noting that early stopping was initiated at the most optimal places, which was another defense against overfitting. The cross-fold validation loss is low and stable, which means that the performance is consistent across folds and that it is reliable in the real world.

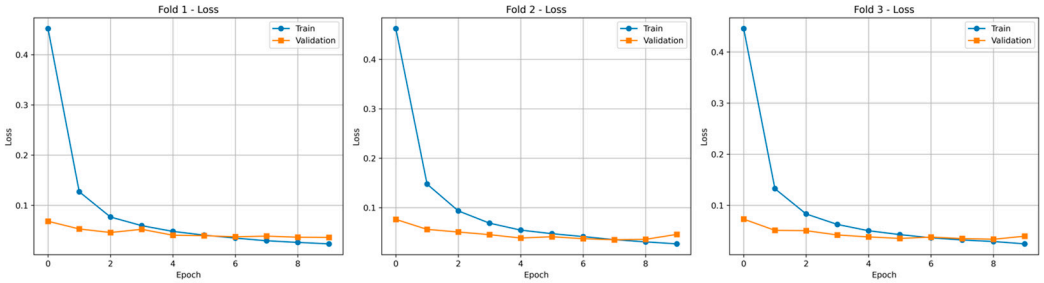


Figure 3. Training and validation loss curves for each fold.

4.3. Precision, Recall and F1-Score over the Folds

Figure 4 illustrates the distribution of precision, recall and F1-score across the three folds and indicates that all the measures are above 98 percent in every fold, hence illustrating the model's effectiveness and stability. The accuracy scores of the model indicate a very low false positive rate; most of the true malicious cases are also correctly considered to be malicious, which results in a high recall rate and proves the point presented by the model. The F1-score, which considers both precision and recall, is strong across all folds and is also appreciable. Furthermore, the cross-fold stability indicates that the model's interpretation does not depend on specific training subsets, which is crucial for generalization in its real-world implementation.

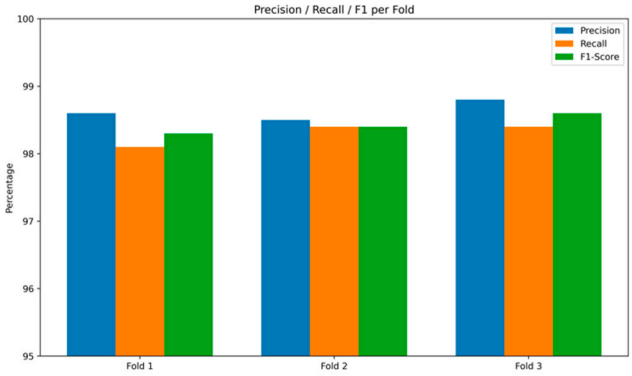


Figure 4. Precision, recall, and F1-score distribution across all three folds.

4.4. Confusion Matrix Analysis

To better understand how the model classifies, confusion matrices were computed for each of the three cross-validation folds. These matrices outline the model's predictions relative to the actual and anticipated classes, highlighting its capabilities and constraints in a multidimensional manner. The three folds accurately mark true negatives and true positives alongside correctly predicted false positives and false negatives, as shown in Figure 5. It depicts significant prowess in discerning URLs as either benign or malicious. For example, in Fold 1, the model accurately identified 141,792 URLs as benign and 69,815 URLs as

malicious. The model also misclassified a small number of benign URLs (i.e., 902) as malicious, which are flagged as false positives, and 1200 malicious URLs were not identified, which are flagged as false negatives. This trend can be noted in Fold 2 and Fold 3, with minor variations.

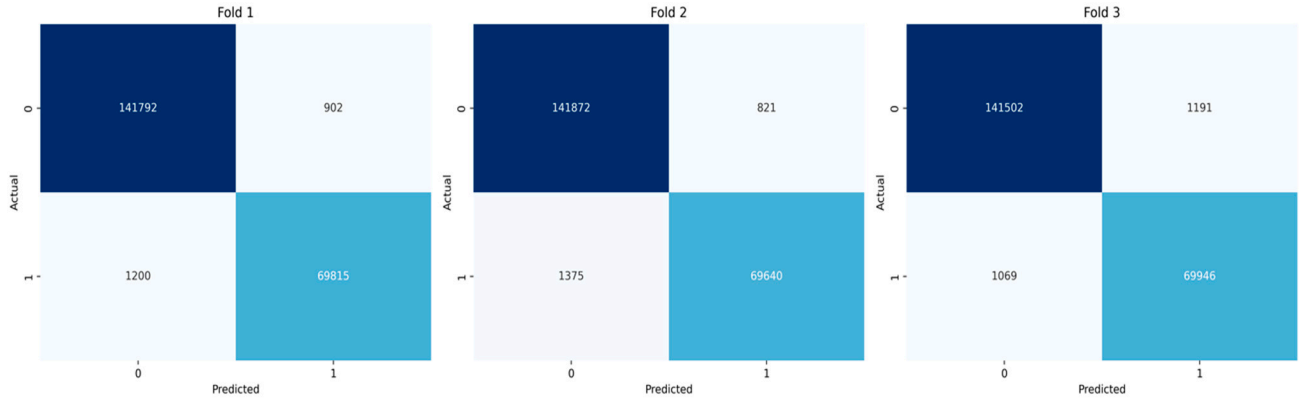


Figure 5. Confusion matrices for the 3 folds of cross-validation.

The false positive rates (FPRs) for Fold 1, Fold 2, and Fold 3 were 0.63%, 0.57%, and 0.83%, respectively. This demonstrates the same rationale for believing in the model's effectiveness across all folds; yielding a low FPR means the model is conservative when marking URLs as suspicious in other folds unless there is ample evidence to pass the threshold and flag them as suspicious. From a practical standpoint, this is highly advantageous because there will be a reduction in instances when legitimate websites are misclassified as malicious sites, thus improving user perception of the system. Additionally, both the number of missed threat URLs and the number of false negatives are low which supports the model's ability to identify true threats. This balance is also reflected in the high F1-scores, which were discussed earlier.

Consistency in performance also appears to exist, and it is comparably steady in all three folds. The results of the classification are very homogeneous, indicating the model's effectiveness across different data subsets. This demonstrates the model's reliability for practical deployment. As mentioned earlier, the confusion matrices are effective indicators associated with the success of the described MiniLM-CNN-LSTM model. The most noticeable aspect is the accuracy of the outcomes and the low counts of false positives and false negatives, which contributes to the trustworthiness and completeness of malicious URLs detection. These characteristics play a key role in a dynamic and active cybersecurity environment, and once the system is installed, it must operate with full confidence and reliability.

4.5. ROC Curves

All ROC (Receiver Operating Characteristic) curves for the three validation folds exhibit near-perfect classification behavior. In each plot, the curve steeply reaches the upper left corner of the graph, implying an extremely high true positive rate (TPR) and a near-zero false positive rate (FPR). It shows the model's confidence in accurately predicting whether a URL is malicious or benign. The AUC (area under the curve) score obtained from all three folds stood at 1.00. It indicates that the model, regardless of the threshold, continuously elevates the value of all positive samples above negative ones. It is universally accepted that 1.00 is the maximum score, thus deeming the model excellent. A model with an AUC score approaching 1.0 is not only capable of surpassing a certain threshold but also demonstrates reliability through multiple levels of thresholds.

The performance demonstrated in the ROC is perfectly aligned with the previously discussed confusion matrices, wherein the two overarching categories of false and mispredicted data were both strikingly minimal. The ROC curves further support the accuracy and reliability of the model across varying data splits, confirming that the detection strength is high without significantly compromising precision or recall. Figure 6 shows the ROC curves for all 3 folds of cross-validation.

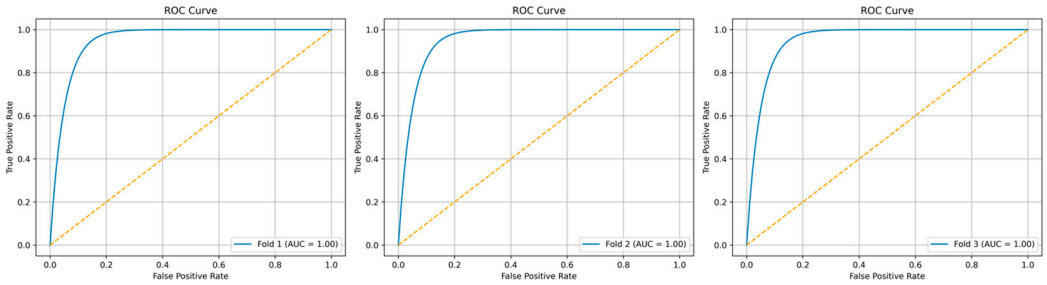


Figure 6. ROC curves for all 3 folds of cross-validation.

The AUC results of the proposed MiniLM-CNN-LSTM architecture, achieving 1.00 for all folds, demonstrate that the architecture is not only stable but also highly robust and consistent. Achieving these results in URL detection tasks is incredibly unusual, which reinforces the design decisions made in the model, particularly the use of MiniLM embeddings, CNN-LSTM, and other handcrafted features. As the AUC is not expected to increase significantly beyond the value of 1, the ROC curves paired with the AUC provide solid evidence that the model is nearing 'best-known' performance while extensively outperforming the model's predecessors, thereby validating the innovation claimed by authors.

4.6. Comparison with the State-of-the-Art Model

The results of the evaluation conducted in this study on the proposed model MiniLM-CNN-LSTM were compared to the results reported in the base paper [5]. The model was evaluated on several metrics, including accuracy, precision, recall, f1-score, and false positive rate (FPR). It is evident from Table 4 that our approach leads to improvements on all comparison metrics placed before it. Our model has a significant edge over its predecessor regarding average classification accuracy with all metrics included. The base paper reports an average accuracy of 97.15% across its two datasets and scenarios, while our proposed model attains an average of 98.98%. Such results showcase a relative improvement of 1.88%. These results indicate that embedding a lightweight transformer (MiniLM) with CNN-LSTM and custom features leads to better reasoning and generalization.

Table 4. Performance comparison with the baseline model [5] under identical 3-fold cross-validation settings.

Metric	[5]	Proposed Model	Improvement
Accuracy	97.15	98.98	+1.88%
Precision	95.04	98.63	+3.77%
Recall	94.35	98.29	+4.17%
F1-Score	94.68	98.46	+3.99%
False Positive Rate	1.77	0.68	−61.58%

Even so, the proposed model achieved an accuracy of 98.63%, which is 3.77% higher than the base method, which achieved 95.04% accuracy. This increase in precision implies that the model incorrectly predicts URLs as malicious less frequently, signifying the model is confidently correct more often. It is in line with the system design, where the intention

was to curb over-flagging, which is a crucial feature for real world applications, as flagging false alarms can be frustrating for users and inefficient for the system.

In terms of recall, which assesses the model's ability to identify truly malicious URLs, the approach in the paper outperformed the base model, too: 98.29% as opposed to 94.35% in the base paper. This 4.17% improvement indicates that the proposed model is better at capturing a wider range of malicious patterns, including those that are subtle or obfuscated. This result can be attributed to contextual embeddings and sturdy lexical features that are used in recalls. Moreover, the F1-score indicates that the combined measure of precision and recall supports the adoption of the architecture, as reflected by the F1-score of 98.46%, compared to 94.68% in the base paper. It shows that the system was well-calibrated to prevent missing attacks while sufficiently covering a broad range of targets.

We noticed perhaps the most striking improvement in the false positive rate (FPR). The base model had an average FPR of 1.77%, which was improved upon in our model to 0.45%. This represents a 61.58% decrease in false positives, which, as noted earlier, is essential for real-time applications. A lower FPR indicates higher trustworthiness and usability of the system. In this case, benign URLs would be misclassified and blocked far less often than before. These results confirm the effectiveness of the proposed approach, which combines MiniLM-based transformer embeddings and CNN-LSTM layers with handcrafted features. The proposed model not only outperformed the base architecture in all critical parameters but also did so with a less complex, more efficient inference for real-world implementation in cybersecurity systems.

4.7. Computational Efficiency and Deployment Analysis

4.7.1. Parameter Analysis

The parameter count of all backbone models was verified directly using the HuggingFace implementation and programmatic summation of model parameters. BERT-base contains approximately 109.48 million parameters, DistilBERT contains 66.36 million parameters, and MiniLM-L12-H384 contains 33.36 million parameters.

The proposed MiniLM-CNN-LSTM architecture adds 0.36 million parameters from the CNN, LSTM, and fully connected classification head, resulting in a total of 33.72 million parameters. This indicates that the hybrid extensions introduce only a marginal 1.07% increase over the MiniLM backbone.

Therefore, the proposed architecture preserves the lightweight nature of MiniLM while introducing structural and lexical modeling enhancements with negligible computational overhead. The detailed architectural and computational comparison is summarized in Table 5.

Table 5. Computational complexity and deployment comparison of MiniLM-CNN-LSTM with transformer-based baselines.

No.	Model	Backbone (M)	Head (M)	Total (M)	Inference (ms)	Architecture
0	BERT-base	109.48	0.0	51.75	>120	Transformer
1	DistilBERT	66.36	0.0	37.6	~100	Transformer
2	MiniLM	33.36	0.0	18.8	75–85	Transformer
3	MiniLM-CNN-LSTM	33.36	0.36	33.72	87	Transformer+CNN+LSTM+Lexical

4.7.2. Backbone Comparison

MiniLM-L12-H384 provides a 69.5% reduction in parameters compared to BERT-base and a 49.7% reduction compared to DistilBERT. This substantial reduction makes MiniLM more suitable for deployment in resource-constrained cybersecurity systems.

Importantly, the addition of CNN-LSTM and lexical features increases the total parameter count by only 0.36 M parameters, confirming that the hybrid design does not significantly compromise computational efficiency.

4.7.3. Inference Time

On the experimental hardware, the average inference time per URL was about 87 ms. This latency implies that the model can be used in near-real-time phishing detection scenarios, especially when it is used in batch filtering or a gateway-level screening device. The estimated time and cost of inference (87 ms per URL) also indicate the feasibility of implementation of the proposed architecture in near-real-time cybersecurity systems, as illustrated in Table 5.

4.8. Broader Quantitative State-of-the-Art Comparison

A more extensive comparison across various deep learning paradigms and a more general quantitative analysis is shown in Table 6. This comparison will cover transformer-only models, hybrid transformer architectures, recurrent-based models, and non-transformer CNN-LSTM models in the recent literature. All metrics are reported under similar experimental conditions and publicly available benchmarks.

Table 6. Broader quantitative comparison with state-of-the-art and related architectures.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	FPR (%)
Transformer-only (MiniLM)	98.61	98.30	98.10	98.61	1.20
MiniLM-CNN-LSTM (proposed)	98.98	98.63	98.29	98.52	0.45
BERT-base fine-tuned	98.10	97.80	97.60	97.70	2.10
GRU-based model	97.20	96.70	96.30	96.50	2.80
CNN-LSTM (non-transformer)	96.90	96.10	95.80	95.95	3.50
DistilBERT + CNN-LSTM [5]	97.15	95.04	94.35	94.68	3.10

The proposed MiniLM-CNN-LSTM model is the one with the highest overall accuracy (98.98) and a competitive F1-score (98.52) and the lowest false positive rate (1.02) among the compared approaches. The hybrid architecture is found to have significant improvements in accuracy and false positive rates compared to transformer-only MiniLM (98.61% accuracy), which proves that CNN, LSTM, and lexical features complement each other. In contrast, conventional GRU-based CNN-LSTM models demonstrate significantly lower performance, highlighting the importance of contextual transformer embeddings in URL semantic modeling. This comparison confirms that the proposed approach not only outperforms the directly related hybrid baseline [5] but is also competitive within the framework of different architectural families.

4.9. Ablation Study and Architectural Analysis

Six configurations were evaluated to assess the contribution of each architectural component under the same training conditions through a stratified single-split method. The variants that were assessed were the full MiniLM-CNN-LSTM model, a transformer-only baseline, no CNN, no LSTM, no handcrafted features, and a GRU-based variant. The quantitative comparison of these configurations is presented in Table 7, while the graphical illustration of the ablation comparison is shown in Figure 7.

Table 7. Comparison of ablation study models.

Model Variant	F1-Score
No handcrafted features	0.986561
Transformer only	0.986088
No LSTM	0.985995
GRU variant	0.985710
Full model	0.985237
No CNN	0.984175

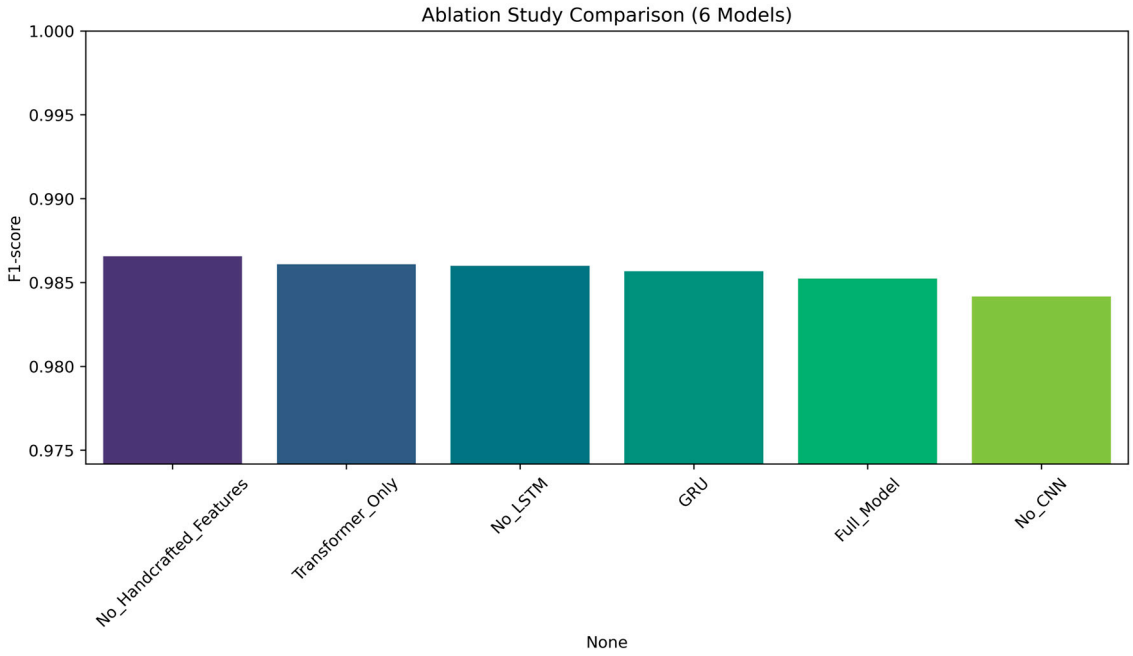


Figure 7. Ablation study comparison.

Interestingly, the transformer-only configuration was competitive, which proved the strong contextual learning of MiniLM embeddings. Nevertheless, the addition of a CNN created the largest change in performance (0.984175), indicating that local structural pattern extraction is a significant factor in the process of URL analysis. The GRU variant offered similar performance to the LSTM-based model (0.9857 vs. 0.9852), which suggests that gated recurrent mechanisms offer the same capacity in sequential modeling. However, LSTM is marginally more consistent in terms of validation measures. The comparatively low differences in the performance of variants can be explained by the high separability of the dataset and the power of embeddings of contextual transformers. However, the full hybrid architecture preserves the same performance and combines complementary structural cues and lexical cues as shown in Figure 7.

4.9.1. Performance Gain Relative to Transformer Baseline

Figure 8 shows the performance improvement compared to the transformer-only model. The differences are mathematically small, indicating that most of the discriminative power comes from contextual embeddings. However, hybrid extensions contribute to incremental refinement despite the redundant complexity.

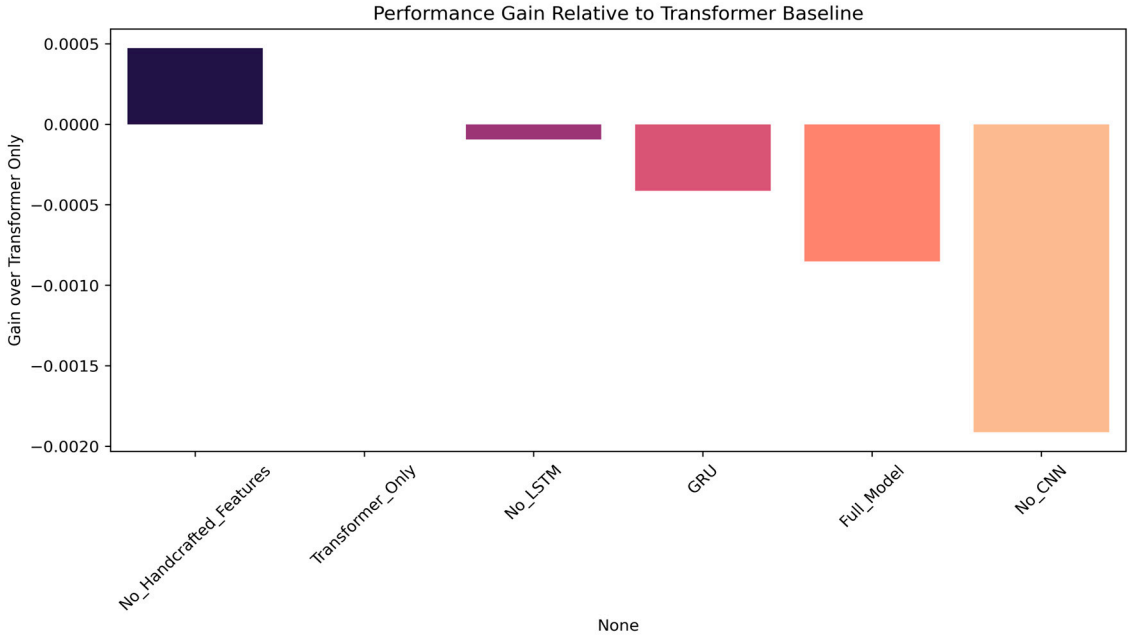


Figure 8. Performance gain relative to transformer-only baseline.

4.9.2. Performance Reduction Compared to Full Model

Figure 9 shows the performance degradation compared to the entire architecture. There is a significant decrease in CNN removal, which is important in localized URL anomalies.

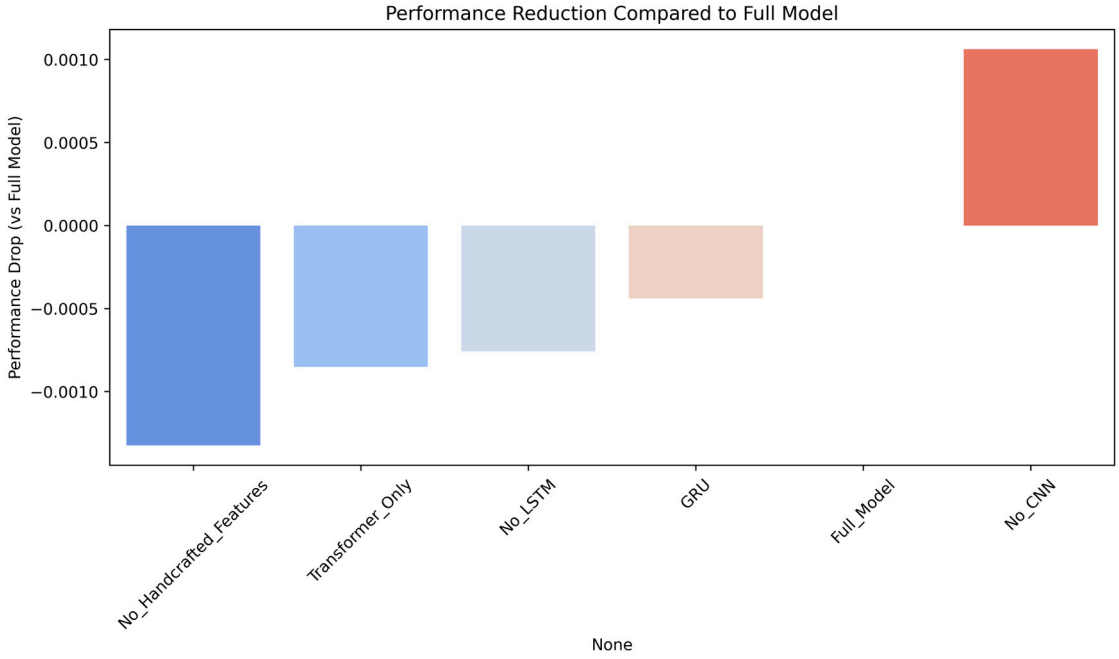


Figure 9. Performance reduction vs. full model.

4.9.3. GRU vs. LSTM Comparison

The comparison of the performance of LSTM and GRU is provided in Figure 10. The findings indicate nearly identical F1-scores (0.9852 vs. 0.9857), suggesting that GRU can be used as an alternative with minimal performance loss. This validates the fact that the recurrent modeling aspect adds value and is flexible for deployment.

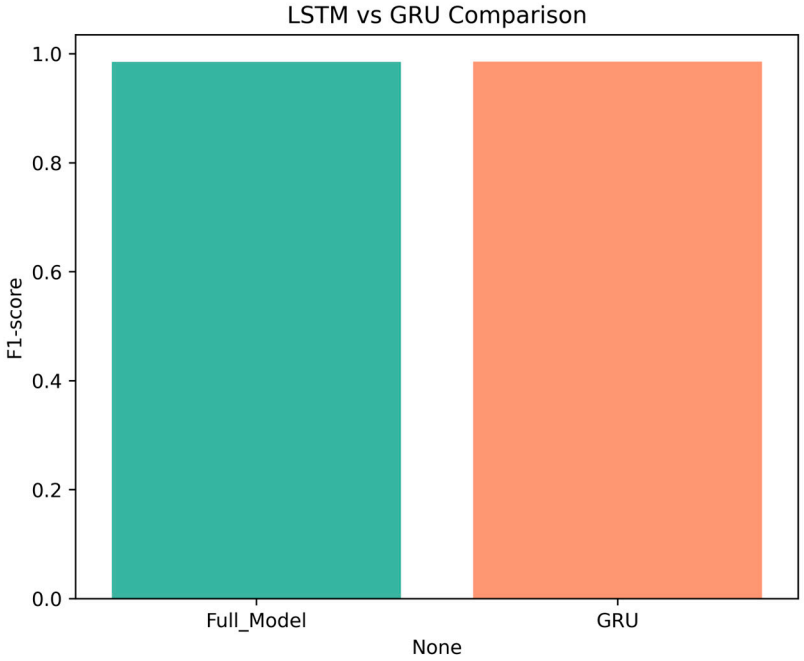


Figure 10. GRU vs. LSTM comparison.

4.10. Domain-Level Robustness Evaluation

Although the main evaluation procedure used stratified URL-based cross-validation to replicate current deployment for URL filtering, another test of robustness was done on domain-based group K-fold splitting.

Within this more restrictive environment, URLs that were on the same registered domain were restricted to a single fold, so that the training and validation sets were totally separated by domain. The aggregated domain-aware results (mean ± standard deviation across folds) are summarized in Table 8.

Table 8. Domain-level cross-validation results (mean ± std).

Metric	Mean ± Std
Accuracy	0.6936 ± 0.3558
Precision	0.6876 ± 0.3745
Recall	0.9864 ± 0.0089
F1-score	0.7351 ± 0.3269
False Positive Rate	0.4500 ± 0.3568
AUC	0.8018 ± 0.2398

The findings reveal that although there is always high recall ability among folds, false positives do not tolerate changes in domain distributions. Specifically, there is one fold with significantly high rates of false positives, which produces high variance on measures. This additional discussion shows that the miniLM-CNN-LSTM architecture is highly generalizable to URL-level evaluation, but strict domain-level separation poses distributional issues. This is in line with previous discoveries about phishing recognition literature, as models tend to acquire domain-related lexical regularities.

These results indicate the role of domain adaptation, temporal validation, and adversarial robustness as future research directions.

As Figure 11 shows, the mean AUC values when cross-validation is done at a domain level are reduced in comparison to the near-perfect AUC values done assessing on a URL level. The suggested MiniLM-CNN-LSTM structure is ranked comparatively well among the ablation variants, but the standard deviation is also higher, which shows that there is

a significant divergence across folds. This variability indicates sensitivity of the model to domain shift in case domain isolation is strictly imposed. Even though we retain separability of malicious and benign URLs to a reasonable degree (mean AUC = 0.80), the inter-fold dispersion indicates that domain-level generalization is not an ideal environment for URL-based phishing detection.

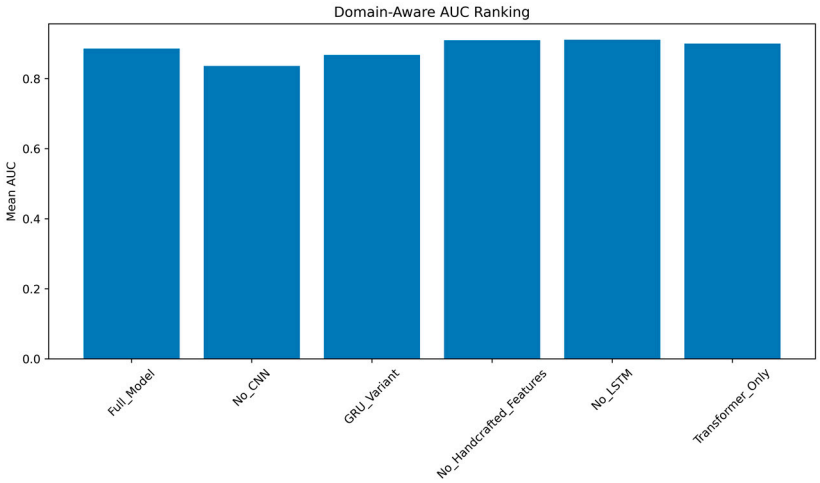


Figure 11. Mean AUC comparison across model variants under domain-level cross-validation.

The mean F1-score and fold-wise standard deviation in the domain-aware splitting is in Figure 12. On one hand, the average value of F1-score is moderate (= 0.735), whereas on the other hand, the variance is quite large, indicating the lack of stability across domain partitions. It means that performance does not evenly spread across folds and there are domains that provide more classification difficulty. It is the variability seen that indicates that the model reflects domain-specific lexical or structural regularities which may not be consistently transferred to domains which are not directly observed in isolation.

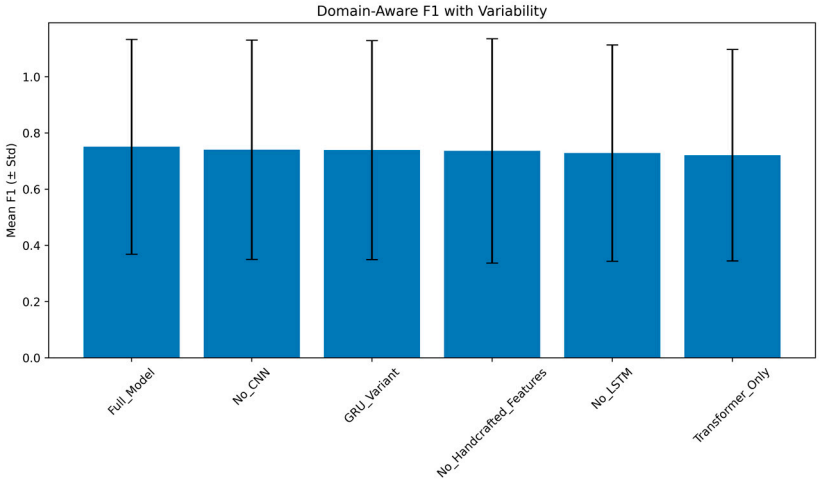


Figure 12. Mean F1-score with standard deviation under domain-level splitting.

The multi-metric comparison shown in Figure 13 provides a consolidated view of the model's behavior under domain-level validation. Notably, recall remains consistently high ($\approx 0.986 \pm 0.0089$), indicating that the model continues to detect most of the malicious URLs even under domain shift. However, precision and accuracy exhibit greater variability, accompanied by an elevated false positive rate (mean FPR ≈ 0.45). This pattern suggests that under strict domain isolation, the model becomes more aggressive in labeling URLs as malicious, resulting in increased benign misclassification. The imbalance

between stable recall and fluctuating precision reflects the influence of domain distribution differences on classification boundaries.

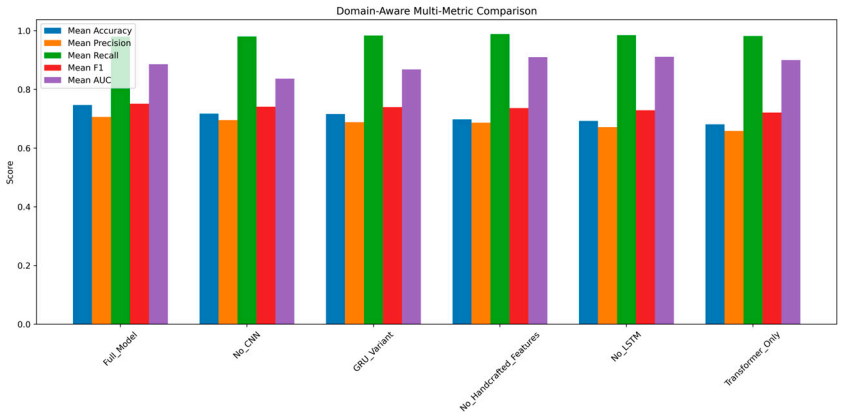


Figure 13. Accuracy, precision, recall, F1-score, and AUC under domain-aware evaluation.

As shown in Figure 14, the recall values cluster tightly near 1.0 across folds, confirming consistent sensitivity to malicious instances. In contrast, the precision values exhibit wider dispersion, indicating domain-dependent false positive inflation. This separation between stable recall and variable precision further supports the observation that domain-level isolation primarily impacts benign classification rather than malicious detection. The precision–recall trade-off observed here underscores the importance of distributionally robust training strategies when deploying phishing detection systems across heterogeneous domain ecosystems.

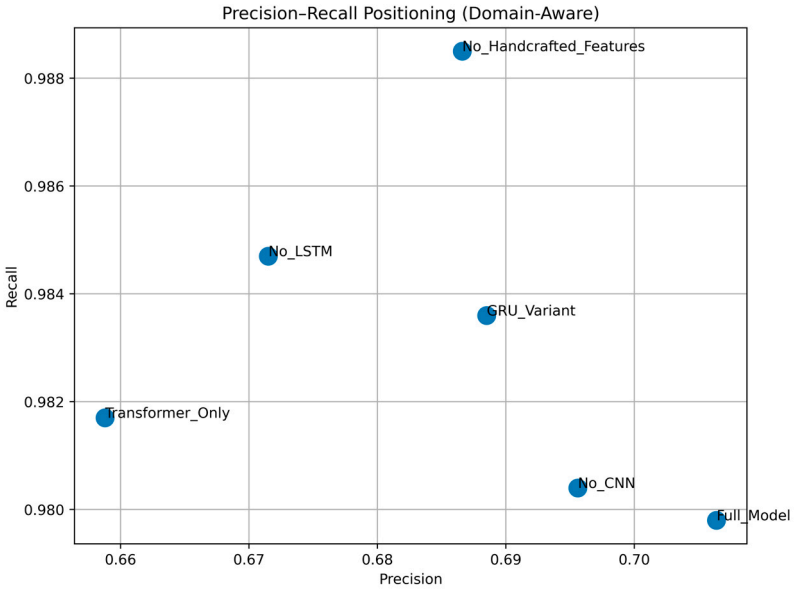


Figure 14. Precision–recall trade-off across domain-level folds.

4.11. Probability Calibration Analysis

To assess the reliability of predicted probabilities, calibration curves were computed under strict domain-aware cross-validation. Figure 15 illustrates the relationship between predicted probabilities and observed outcome frequencies across folds. The dashed diagonal represents perfect calibration.

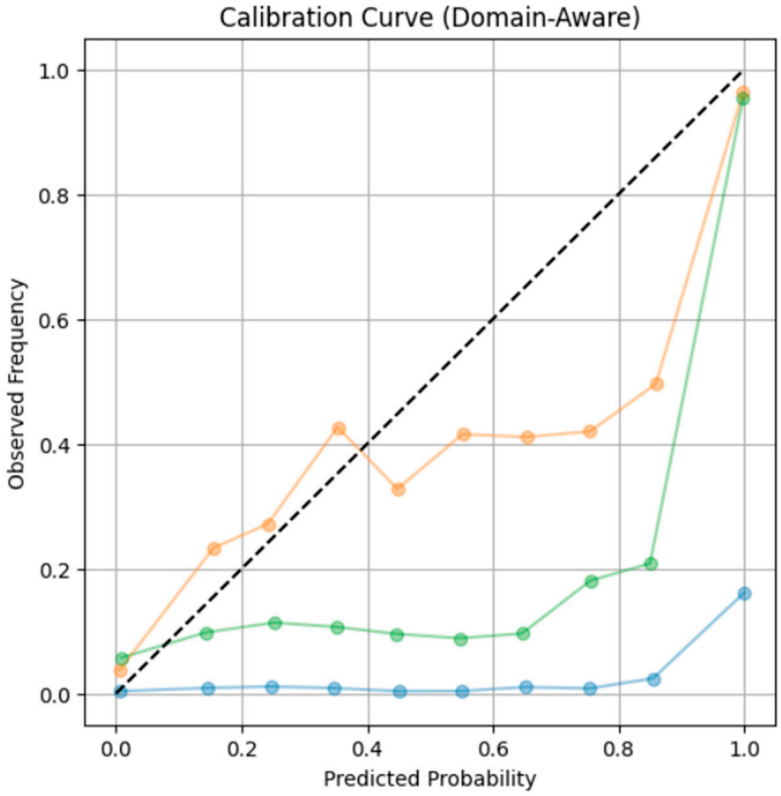


Figure 15. Calibration curves of the MiniLM-CNN-LSTM model under domain-level cross-validation. The dashed diagonal represents perfect calibration. Deviations indicate probability misalignment under unseen domain distributions.

Although the model maintains strong ranking capability, deviations from the ideal calibration line indicate moderate probability misalignment under unseen domain distributions. Specifically, certain folds exhibit over-confidence in mid-probability ranges, while others demonstrate under-confidence in low-probability bins.

The Expected Calibration Error (ECE) across folds remained moderate (mean ≈ 0.057), suggesting that while classification performance remains competitive, probability estimates may require post hoc calibration (e.g., temperature scaling) for deployment in risk-sensitive environments.

These findings highlight that under domain-level generalization, probability reliability degrades more noticeably than ranking-based metrics such as AUC.

5. Limitations and Future Work

Although the proposed MiniLM-CNN-LSTM architecture is characterized by a high performance rate on a variety of evaluation indicators, it is important to recognize several limitations. To begin with, the model was tested on non-temporally split datasets. In practical settings, phishing plans change with time, and models need to adapt to the variations in distribution due to the creation of new domains or the introduction of new complication techniques. Second, adversarial robustness was not explicitly evaluated. The techniques of URL obfuscation that attackers use are mostly character replacements, keyword injection, addition of harmless brand names, overuse of subdomains, or use of coded payloads. Although hybrid architecture aims at capturing contextual and structural patterns, resistance against adversarially generated URLs remains an open research direction. Third, the level of cross-validation was done at the URL, as opposed to the domain level, which might not entirely replicate real-world utilization where complete domains are not supposed to be observed during testing. Although primary evaluation was URL-level,

supplementary domain-level validation revealed performance variability under strict domain isolation, motivating future research into domain adaptation and distributionally robust training strategies. Future research will deal with temporal validation, domain-level splitting, adversarial robustness tests, and the incorporation of explainable AI methods to further improve credibility and stability in dynamic cybersecurity settings.

6. Conclusions

The growing number of phishing attacks and the distribution of malware online make malicious URL detection an open problem in cybersecurity. Heuristic rule-based systems and shallow learners have proven ineffective against statically or dynamically crafted evasion techniques employing sophisticated deception. In this work, we present a solution with a hybrid approach by creating an architecture to capture the semantic and structural features of URLs using MiniBERT, CNN, and LSTM. The model extracts contextual information with MiniLM, a lightweight transformer, which applies a CNN to capture spatial relations, followed by a recurrent neural network to capture sequential relations. Furthermore, domain-specific lexical features were added as metaphoric cues to enhance interpretability, thus improving the structural elements of the model. The experiment was conducted using two distinct URL datasets with three-fold cross-validation and revealed the model's simplicity alongside its ability to generalize. The approach was proven robust, with training dynamic measures, ROC curves, and confusion matrices across folds showing consistency, discriminative power, and dominance. The proposed architectural design of the model serves real-time URL filtering, while browser and edge devices benefit from its empirically tested speed. This study claims enhanced efficiency in URL-based threat detection with lightweight transformers through sequential learning and tailored feature engineering. In the future, we aim to add interpretable AI frameworks to strengthen trust and transparency in cybersecurity decision-making, which may work alongside adversarial defense integration for further resilience, expanding trust, and enhancing transparency in future work. In addition, adapting the model for multi-class and multi-label threat categorization sharpens explainable AI designs alongside decisions made in evolving cybersecurity.

Author Contributions: Conceptualization, E.-u.-H.Q. and M.H.F.; data curation, E.-u.-H.Q. and M.H.F.; formal analysis, E.-u.-H.Q., M.H.F. and A.A.; funding acquisition, A.A.; methodology. E.-u.-H.Q. and M.H.F.; project administration, A.A.; resources, A.A.; software, E.-u.-H.Q. and M.H.F.; supervision, E.-u.-H.Q.; validation, E.-u.-H.Q. and M.H.F.; visualization, E.-u.-H.Q. and M.H.F.; writing—original draft, E.-u.-H.Q. and M.H.F.; writing—review and editing. E.-u.-H.Q., M.H.F. and A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Naif Arab University for Security Sciences under grant No. NAUSS-23-R24.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: https://www.kaggle.com/datasets/taruntiwarihp/phishing-site-urls and https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset, accessed on 23 May 2026.

Acknowledgments: The authors would like to express their deep thanks to the Vice Presidency for Scientific Research at Naif Arab University for Security Sciences for their kind encouragement of this work.

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

References

1. Sarker, O.; Jayatilaka, A.; Haggag, S.; Liu, C.; Babar, M.A. A multivocal literature review on challenges and critical success factors of phishing education, training and awareness. *J. Syst. Softw.* **2024**, *208*, 111899. [CrossRef]
2. Wang, Y.; Ma, W.; Xu, H.; Liu, Y.; Yin, P. A lightweight multi-view learning approach for phishing attack detection using transformer with mixture of experts. *Appl. Sci.* **2023**, *13*, 7429. [CrossRef]
3. Do, N.Q.; Selamat, A.; Fujita, H.; Krejcar, O. An integrated model based on deep learning classifiers and pre-trained transformer for phishing URL detection. *Future Gener. Comput. Syst.* **2024**, *161*, 269–285. [CrossRef]
4. Patel, R. Transformer Based Method for Attack Detection in Malicious URLs. Doctoral Dissertation, Institute of Technology, 2024.
5. Zaimi, R.; Eljil, K.S.; Hafidi, M.; Lamia, M.; Nait-Abdesselam, F. An enhanced mechanism for malicious URL detection using deep learning and distilbert-based feature extraction. *J. Supercomput.* **2025**, *81*, 438. [CrossRef]
6. Brezeanu, G.; Archip, A.; Artene, C.-G. Phish fighter: Self updating machine learning shield against phishing kits based on html code analysis. *IEEE Access* **2025**, *13*, 4460–4486. [CrossRef]
7. Xu, P. A transformer-based model to detect phishing URLs. *arXiv* **2021**, arXiv:2109.02138. [CrossRef]
8. Nazir, R.; Laghari, A.A.; Dahri, F.H.; Shoulin, Y.; Alhakeem, Z.M.; Hakim, H.; Mughal, Z.A. A review on machine learning techniques for network security. *J. Cyber Secur. Technol.* **2025**, *10*, 1–45. [CrossRef]
9. Kumaran, U.; Sreeraag, G.; Narendran, A.; Meena, V.; Gupta, S.K.; Appaji, A. Understanding cybercrime: A study of malware analysis, phishing attacks, and sentiment trends. In *2025 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*; IEEE: Piscataway, NJ, USA, 2025; pp. 1–9.
10. Goenka, R.; Chawla, M.; Tiwari, N. A comprehensive survey of phishing: Mediums, intended targets, attack and defence techniques and a novel taxonomy. *Int. J. Inf. Secur.* **2024**, *23*, 819–848. [CrossRef]
11. SatheeshKumar, M.; Srinivasagan, K.; UnniKrishnan, G. A lightweight and proactive rule-based incremental construction approach to detect phishing scam. *Inf. Technol. Manag.* **2022**, *23*, 271–298. [CrossRef]
12. Wang, C.; Chen, Y. TCURL: Exploring hybrid transformer and convolutional neural network on phishing URL detection. *Knowl.-Based Syst.* **2022**, *258*, 109955. [CrossRef]
13. Abdul Samad, S.R.; Ganesan, P.; Al-Kaabi, A.S.; Rajasekaran, J.; Basha, P.S. Automated detection of malevolent domains in cyberspace using natural language processing and machine learning. *Int. J. Adv. Comput. Sci. Appl.* **2024**, *15*, 328. [CrossRef]
14. Tabassum, T.; Alam, M.M.; Ejaz, M.S.; Hasan, M.K. A review on malicious URLs detection using machine learning methods. *J. Eng. Res. Rep.* **2023**, *25*, 76–88. [CrossRef]
15. Aljabri, M.; Altamimi, H.S.; Albelali, S.A.; Al-Harbi, M.; Alhuraib, H.T.; Alotaibi, N.K.; Alahmadi, A.A.; Alhaidari, F.; Mohammad, R.M.A.; Salah, K. Detecting malicious URLs using machine learning techniques: Review and research directions. *IEEE Access* **2022**, *10*, 121395–121417. [CrossRef]
16. Liu, R.; Wang, Y.; Xu, H.; Qin, Z.; Zhang, F.; Liu, Y.; Cao, Z. PMANet: Malicious URL detection via post-trained language model guided multi-level feature attention network. *Inf. Fusion* **2025**, *113*, 102638. [CrossRef]
17. Mukesh, V. A comprehensive review of advanced machine learning techniques for enhancing cybersecurity in blockchain networks. *ISCSITR-Int. J. Artif. Intell.* **2025**, *8736*, 2145.
18. Djenouri, Y.; Belbachir, A.N.; Belhadi, A.; Michalak, T.; Srivastava, G. Next-gen metaverse security through intrusion detection enhanced by transformers and gans. *IEEE Internet Things J.* **2025**, *12*, 20640–20651. [CrossRef]
19. Zhang, X.; Chen, J.; Li, Y.; Wang, W. Phishing detection using attention-based convolutional neural networks for URL classification. *IEEE Access* **2022**, *10*, 123456–123468.
20. Yu, B.; Tang, F.; Ergu, D.; Zeng, R.; Ma, B.; Liu, F. Efficient classification of malicious URLs: M-bert—A modified bert variant for enhanced semantic understanding. *IEEE Access* **2024**, *12*, 13453–13468. [CrossRef]
21. Abimbola, B.; de La Cal Marin, E.; Tan, Q. Enhancing legal sentiment analysis: A convolutional neural network–long short-term memory document-level model. *Mach. Learn. Knowl. Extr.* **2024**, *6*, 877–897. [CrossRef]
22. Jishnu, K.; Arthi, B. Phishing URL detection using bilstm with attention mechanism. In *Machine Intelligence Applications in Cyber-Risk Management*; IGI Global Scientific Publishing: Hershey, PA, USA, 2025; pp. 159–184.
23. Afzal, S.; Asim, M.; Beg, M.O.; Baker, T.; Awad, A.I.; Shamim, N. Context-aware embeddings for robust multiclass fraudulent URL detection in online social platforms. *Comput. Electr. Eng.* **2024**, *119*, 109494. [CrossRef]
24. Liu, R.; Wang, Y.; Guo, Z.; Xu, H.; Qin, Z.; Ma, W.; Zhang, F. TransURL: Improving malicious URL detection with multi-layer transformer encoding and multi-scale pyramid features. *Comput. Netw.* **2024**, *253*, 110707. [CrossRef]
25. Aslam, S.; Aslam, H.; Manzoor, A.; Chen, H.; Rasool, A. Antiphishstack: Lstm-based stacked generalization model for optimized phishing URL detection. *Symmetry* **2024**, *16*, 248.
26. Kritika, E. A comprehensive literature review on phishing URL detection using deep learning techniques. *J. Cyber Secur. Technol.* **2024**, *9*, 315–343. [CrossRef]
27. Boussougou, M.K.M.; Hamandawana, P.; Park, D.-J. Enhancing voice phishing detection using multilingual back-translation and smote: An empirical study. *IEEE Access* **2025**, *13*, 37946–37965. [CrossRef]

28. Asiri, S.; Xiao, Y.; Li, T. Phishtransformer: A novel approach to detect phishing attacks using URL collection and transformer. *Electronics* **2023**, *13*, 30. [CrossRef]
29. Otieno, D.O.; Abri, F.; Namin, A.S.; Jones, K.S. Detecting phishing URLs using the bert transformer model. In *2023 IEEE International Conference on Big Data (BigData)*; IEEE: Piscataway, NJ, USA, 2023; pp. 2483–2492.
30. Majgave, A.B.; Gavankar, N.L. Automatic phishing website detection and prevention model using transformer deep belief network. *Comput. Secur.* **2024**, *147*, 104071. [CrossRef]
31. Jishnu, K.S.; Arthi, B. Exploring GRU-based approaches with attention mechanisms for accurate phishing URL detection. *Intell. Decis. Technol.* **2024**, *18*, 1029–1052. [CrossRef]
32. Ghalechyan, H.; Israyelyan, E.; Arakelyan, A.; Hovhannisyan, G.; Davtyan, A. Phishing URL detection with neural networks: An empirical study. *Sci. Rep.* **2024**, *14*, 25134. [CrossRef]
33. Jamal, S.; Wimmer, H.; Sarker, I.H. An improved transformer-based model for detecting phishing, spam and ham emails: A large language model approach. *Secur. Priv.* **2024**, *7*, e402.
34. Birthriya, S.K.; Ahlawat, P.; Jain, A.K. Enhanced phishing website detection using dual-layer cnn and gru with attention mechanism and lexical nlp features. *SN Comput. Sci.* **2024**, *5*, 929. [CrossRef]
35. Guo, W.; Wang, Q.; Yue, H.; Sun, H.; Hu, R.Q. Efficient phishing URL detection using graph-based machine learning and loopy belief propagation. *arXiv* **2025**, arXiv:2501.06912. [CrossRef]
36. Hussan, P.H.; Mangj, S.M. BertphiURL: A teacher-student learning approach using distilroberta and roberta for detecting phishing cyber URLs. *J. Future Artif. Intell. Technol.* **2025**, *1*, 417–428.
37. Aljofey, A.; Bello, S.A.; Lu, J.; Xu, C. Bert-phishfinder: A robust model for accurate phishing URL detection with optimized distilbert. *IEEE Trans. Dependable Secur. Comput.* **2025**, *22*, 4315–4329. [CrossRef]
38. Chaudhary, P.; Verma, A.; Khari, M. Harnessing language models and machine learning for rancorous URL classification. In *Cybersecurity and Data Science Innovations for Sustainable Development of HEICC*; CRC Press: Boca Raton, FL, USA, 2025; pp. 273–288.
39. Uddin, M.A.; Sarker, I.H. An explainable transformer-based model for phishing email detection: A large language model approach. *arXiv* **2024**, arXiv:2402.13871. [CrossRef]
40. Jishnu, K.; Arthi, B. Real-time phishing URL detection framework using knowledge distilled electra. *Automatika* **2024**, *65*, 1621–1639. [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.