computers

MDPI

Article

Phishing URL Detection Using Transformer-Based Architecture and Contextual Content Features

Emad-ul-Haq Qazi [1,2,*], Muhammad Hamza Faheem [1,3] and Abdulrazaq Almorjan [2,3]

[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, Riyadh 14812, Saudi Arabia
[3] Center of Cybercrimes and Economic Crimes, Naif Arab University for Security Sciences, Riyadh 14812, Saudi Arabia
* Correspondence: qabdulrab@nauss.edu.sa

Abstract

Phishing sites are increasingly causing harm to consumers, commercial enterprises, and the online infrastructure. Online safety is dependent on how well these evil sites can be detected in time and correctly. A number of solutions that exist are based on lexical, token features, or structural hints. Although useful to a certain degree, these methods tend to lose more contextual meaning in URLs. This paper presents SemanticPhishNet, a hybrid detection system that uses semantic knowledge to detect phishing attacks by utilizing semantic knowledge via a transformer-based system to process HTML documents and URL information to produce accurate and efficient phishing detections. The architecture uses MiniLM (identical type as distillbert) to obtain contextual embeddings of cleaned HTML and augmented text of URLs and a simple dense classifier to perform effective binary classification. A stratified three-way split of data was used to evaluate the model with real-world obfuscation patterns like replacement of "http" by "hxxp". The experimental findings show that SemanticPhishNet has high performance in various measures, outperforming other state-of-the-art models in accuracy, recall and generalization ability. We conduct experiments on cross-validation and external validation with independent data. The framework exhibits good performance (96–97% cross-validation accuracy) and external evaluation demonstrates realistic generalization (67% accuracy), albeit revealing the difficulties of domain shift in phishing. The proposed model performs better than many of the existing models in the real world. The confusion matrices and ROC analysis indicate that the phishing and benign classes are consistently separated in both the validation and test sets. The findings show that the suggested model is efficient, stable, and scalable to the present-day phishing detection. The paper stresses the importance of appropriate evaluation techniques, such as leakage-aware splits and cross-dataset evaluation.

Keywords: deep learning; transformer models; feature fusion; hybrid model; MiniLM; phishing detection; semantic embeddings; URL classification

Check for updates

Academic Editors: Phivos Mylonas, Katia Lida Kermanidis and Manolis Maragoudakis

Received: 10 February 2026
Revised: 13 May 2026
Accepted: 14 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

Phishing is one of the most prevalent and harmful types of cyberattacks [1,2]. It is widely employed to trick people into giving sensitive data like the login details, banking details, or personal identifiers. These attacks tend to be based on the fraud of trusted websites or services through manipulation of URLs or webpage contents [3,4].

With the ever-changing complexity and frequency of phishing, conventional methods of detection cannot maintain high accuracy and scalability. Early identification of phishing attack is necessary to minimize financial losses, avoid data breach, and safeguard user confidence in online platforms [5]. The main vectors of phishing attacks are URLs. Attackers often develop malicious.

Web addresses that are quite similar to real web addresses [6]. Among the common techniques are misleading subdomains, abnormal parameter constructs, numeric IPs or coded paths [7–9]. Such manipulations are specially developed to overcome the static filters and deceive the human users. This renders rule-based or blacklist-based systems ineffective in handling new phishing campaigns [10–12]. These systems are largely reliant on patterns known or lists curated and therefore, they are not useful in zero-day phishing.

To close this gap, a number of Machine Learning (ML) solutions have been induced in the last 10 years. These models are based on a mixture of lexical, statistical and structural features derived out of URLs [13–15]. Other techniques also involve HTML content and JavaScript behavior, however, gathering this data would involve complete page rendering, which is not always practical [16]. Deep Learning (DL) methods are promising to learn features of raw inputs automatically [17,18]. Convolutional and recurrent network-based models have been employed to learn local and sequential patterns to identify phishing URLs [19–21]. Nevertheless, most of these models continue to use superficial clues and do not reflect deeper contextual meaning that is hidden in URLs.

Recent developments in Natural Language Processing (NLP) have seen emergence of transformer-based models that have the capability of rich semantics information [22–24]. The high-capacity to model long-range dependencies and contextual relationships has demonstrated good performance of transformers in a broad spectrum of classification tasks. When it comes to phishing detection, there is a potential that such models can learn the intent or purpose of a URL on top of its structure. Together with constructed properties which reflect domain heuristics, a transformer-enhanced model can give a more holistic view of phishing behavior [25,26].

This paper proposes SemanticPhishNet, a hybrid DL framework designed to detect phishing URLs using both semantic and structural information. The model uses a pre-trained transformer to extract contextual embeddings from raw URLs. These embeddings are then combined with handcrafted features that capture syntactic properties of the URL. The combined input is processed through a deep neural network that includes convolutional and recurrent layers, allowing the model to learn spatial and temporal dependencies. The design aims to improve both generalization and interpretability, while avoiding the complexity of HTML-based or browser-dependent solutions. The main contributions of this paper are summarized as follows:

- A phishing detection system, SemanticPhishNet, which derives semantic representations of cleaned HTML tags and URL text with a lightweight transformer network.
- While both URL and HTML content were considered, they are treated as independent textual instances rather than jointly fused modalities.
- A lightweight neural architecture is employed by integrating Distilled Bidirectional Encoder Representations from Transformers (DistilBERT) as the semantic encoder with a dense neural classifier to ensure high detection accuracy and efficiency.
- Augmented data is generated by applying obfuscation-aware transformation (replacing “http” with “hxxp”) to simulate real-world phishing evasion patterns and enhance robustness.
- HTML content is preprocessed using BeautifulSoup to extract semantic text, while tokenization is performed using the DistilBERT tokenizer with a maximum token length of 512.

- The model is trained and validated using a stratified 80-10-10 split to preserve label distribution, and performance is evaluated across validation and test sets.
- Comparative evaluation is conducted against multiple recent state-of-the-art approaches, supported by confusion matrices, ROC curves, and metric-based tables.

The rest of the paper encompasses related work in Section 2, methodology in Section 3, experimentation setup in Section 4, results in Section 5, and the conclusion in Section 7.

2. Related Work

Opara et al. [27] proposed WebPhish, an end-to-end deep learning model that detects phishing web pages using raw URL and HTML content without any manual feature engineering. Their model achieved high accuracy using character and word-level embeddings with convolutional layers. Nevertheless, WebPhish is hindered by its high computational cost, making it more resource-intensive than radiate machine learning techniques. Haq et al. [28] developed a phishing URL detection system based on deep learning techniques using a one-dimensional convolutional neural network (CNN). That said, compared to RNN or transformer-based architectures, this approach may be less effective at capturing complex hierarchical features due to long-range dependencies within the data. Ref. [29] advanced the incorporation of URL parameters integrated with textual semantics for phishing email detection. As promising as this approach is, their review highlighted the susceptibility of these models to adversarial examples specifically tailored to hijack tokenized input sequences designed to manipulate sequence inputs.

In the same way, Tamal et al. [30] proposed an optimal feature vectorization alongside other approaches to improve detection accuracy. However, there is a risk of benchmarking performance against more sophisticated content-based phishing websites that mimic legitimate site behavior. Elkholy et al. [31] proposed an efficient phishing detection framework that uses a hybrid approach to machine learning by integrating Logistic Regression, Support Vector Machines, and Decision Trees through a voting mechanism. Their system improves focus by center feature selection toward classification-URL-based features and improving Random Forest feature importance analysis classification accuracy. Still, their approach is limited to handcrafted URL features and lacks semantic contextual embeddings, which may hinder the detection of sophisticated novel phishing attacks.

Barik et al. [32] designed an EGSO-CNN model for phishing URL detection by applying deep learning with Enhanced Grid Search Optimization (EGSO) to feature extraction using Variational Autoencoders (VAE). They also introduced an updated phishing dataset to enhance the benchmarks' out-of-date constraints. However, it is overly reliant on extensive preprocessing coupled with optimization tuning. Zaimi et al. [33] designed a framework for detecting phishing URLs that fuses semantic feature extraction using DistilBERT with deep learning classification through hybrid CNN-LSTM models. However, the transformer embedding reliance on this approach burdens the system, increasing complexity for real-time or resource-limited scenarios. Setu et al. [34] developed RSTHFS, a Rough Set Theory-based Hybrid Feature Selection method for improving machine learning-based phishing website detection. Their approach attained an average accuracy of 95.48% across multiple datasets while limiting the number of features, computational runtime, and model efficiency, outrunning the competition. Still, conventional ML classifiers and handcrafted features are less flexible than contemporary deep learning or semantic approaches.

Nayak et al. [35] implemented a phishing detection framework that integrated advanced feature engineering with machine learning, utilizing DNNs, Wide and Deep, and TabNet models. Their method improved the detection accuracy to 94.46% after narrowing down the feature set from 111 to 14, alongside the introduction of an innovative "anti-phishing score" metric that optimally balances false positive and false negative rates.

Despite the advanced outcomes of achieving the goals, the model is bound to extensive feature engineering and hyperparameter tuning, limiting its adaptability to evolving phishing strategies rapidly. Gałka et al. [36] proposed an aggregation-based ensemble learning model for phishing attack detection, emphasizing minimizing the false positive rate and maximizing the true positive rate. Their self-tuning ensemble incorporates multiple aggregation functions along with an incremental learning approach that was evaluated on a FreshMail dataset. Although their approach reported statistically significant improvements over traditional neural network models at some TPR cutoffs, as Groudlewski et al. noted, the optimization based on handcrafted aggregation strategies and the specific dataset used may hinder generalizability to other phishing contexts.

Patni et al. [37] implemented a phishing website detection system that uses Natural Language Processing (NLP) features with machine learning models. Their method retrieves and analyzes linguistic elements from queuing web pages' contents like anchor text, meta descriptions, and home page titles. It employs sentiment analysis techniques, keyword matching, and phrase frequency to bolster sensitivity to phishing websites. Nonetheless, the system is innovative, but its overall effectiveness is limited by the abundance and richness of text content, which makes it less effective against minimalistic or highly obfuscated phishing sites. Hamidi and Sayah [38] developed a hybrid stacking-based model for phishing URL detection, integrating Extreme Gradient Boosting (XGBoost) and Random Forest as base classifiers with Logistic Regression as the meta-learner. Their method achieved better accuracy and precision than the standalone models due to the ensemble effect, but still relied on handcrafted features derived from URLs. Such dependence may hinder adaptability to more sophisticated phishing attacks that evolve structurally. Muskan et al. [39] proposed an optimized hybrid feature selection approach for suspicious URL detection in IoT frameworks, combining a filter-based technique with a Genetic Algorithm (GA) to improve feature selection outcomes.

Nevertheless, although their hybrid feature approach optimized lightweight deployment, it will likely face challenges with multi-layered real-time phishing attacks where content dynamically shifts beyond the static lexical and page features. Gupta et al. [40] created a hybrid machine learning system designed specifically for spam detection based on features derived from previous methods, aiming to resolve the accuracy issues caused by older predictive systems. Their method achieved greater accuracy than previous studies due to improved feature engineering and several machine learning classifiers. However, the model's reliance on fixed feature sets will require extensive reworking to adapt to phishing URLs and websites' complex, dynamic environments. Al Ogaili et al. [41] proposed PhishNetVAE, which is built around a fully integrated phishing detection framework that uses a VAE to extract latent features in an unsupervised manner and employs a DNN classifier for the URL phishing classification. The model, however, may not perform well in real-time or resource-limited settings due to heavy computational requirements for training the VAE.

The discussed literature identifies a progression from initial methods of phishing detection using traditional machine learning to more sophisticated methods employing deep learning and hybrid models. Earlier systems were centered on ensemble learning and the construction of features based on URLs, and while they performed adequately, they were not adaptable or scalable. More recent works, such as those utilizing DistilBERT, Variational Autoencoders (VAE), and advanced feature selection techniques, show marked improvement in the accuracy and robustness of detection (see Table 1). However, this often comes at the increased cost of computational complexity, constraining their use in practical, real-time, or resource-limited scenarios. All in all, the pattern is one of increasing detection

performance at the expense of system efficiency, underscoring the need for detection models that are lightweight yet rich in semantics.

Table 1. Comparison of Related Work Approaches for Phishing Detection.

Ref.	Proposed Method	Key Contribution	Limitation
[27]	WebPhish (End-to-end DL with raw URL + HTML embeddings)	High accuracy with character- and word-level embeddings	High computational cost
[28]	1D CNN on URLs	Lightweight deep model for URL-based detection	Struggles with long-range dependencies
[29]	Review on URL-text semantic integration	Emphasized adversarial vulnerability in URL-text models	Models vulnerable to adversarial attacks
[30]	Optimal Feature Vectorization (URL-only)	Improved detection using URL structural features	No content-based feature analysis
[31]	Hybrid ML Ensemble (SVM, LR, DT)	Feature importance optimization with RF voting	Limited to handcrafted URL features
[32]	EGSO-CNN with VAE Features	Optimization using EGSO and VAE-enhanced feature extraction	High preprocessing and tuning effort
[33]	DistilBERT + CNN-LSTMHybrid	Semantic-rich feature extraction with transformers	Computationally intensive
[34]	Rough Set Theory Hybrid Selection (RSTHFS)	Feature reduction with increased efficiency	Limited adaptability to deep learning approaches
[35]	DNN, Wide and Deep, TabNet with Feature Selection	Novel Anti-Phishing Score and robust feature selection	Dependency on large-scale manual features
[36]	Aggregation-based Ensemble (Incremental Learning)	Lower FPR with incremental learning strategies	Dataset-specific optimization required
[37]	NLP-based Phishing Detection (Anchor texts, Meta descriptions)	Context-based linguistic feature analysis	Limited performance on minimalistic phishing
[38]	Stacking Ensemble (XGBoost + RF + LR)	Improved precision viahybriden-sembling	Still relies on handcrafted features
[39]	GA-Enhanced Hybrid Feature Selection	IoT-specific lightweight phishing detection	May struggle against fast-evolving attacks
[40]	Hybrid ML for Feature-based Spam Detection	Enhanced precision via hybrid feature engineering	Static features limit phishing adaptability
[41]	PhishNetVAE (VAE + DNN Hybrid)	High detection accuracy via latent feature learning	Heavy resource requirement for VAE training

3. Methodology

This segment introduces the suggested phishing URL detection framework, which is aimed at providing high-quality performance, scalability, and high data leakage prevention. The proposed system concentrates on the analysis of URLs exclusively, as opposed to the multi-modal methods that have been used previously, which allows real-world implementation and avoids the risks of having multiple semantic content in the training and evaluation sets. The pipeline is generally divided into four key steps, including building of datasets, preprocessing and feature extraction, splitting and augmentation of data, and training a model based on a hybrid transformer-based architecture. The processed text is then tokenized and passed into the DistilBERT (MiniLM) model, which extracts contextual embeddings. These embeddings are fed into a dense neural network for final classification. The integration of semantic embeddings with a lightweight architecture ensures both efficiency and high accuracy in phishing detection. Figure 1 shows the pipeline for the proposed SemanticPhishNet architecture.

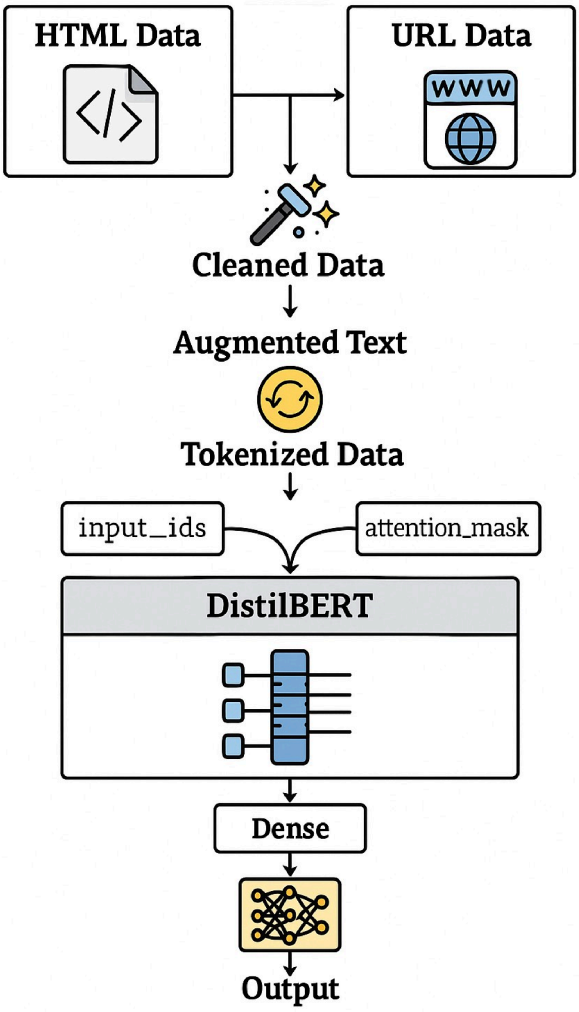


Figure 1. Proposed SemanticPhishNet Pipeline.

Algorithm 1 outlines the complete workflow of the proposed SemanticPhishNet frame-work. The raw input data d_i is first cleaned using a transformation function T and augmented via a replacement function R to produce $C\tilde{}i$. The resulting dataset D' is split into training (D_1), validation (D_2), and test (D_3) subsets. Each cleaned text sample C_j is tokenized using the encoder E into input IDs x_j and attention masks m_j of fixed length ℓ. During training, batches of (x_b, m_b, y_b) are passed through the transformer model M to obtain hidden states h_b, from which the CLS embedding h_{CLS} is extracted. This embedding is classified using C to predict $\hat{y}_b$, and the model is optimized using binary cross-entropy loss L and learning rate η for E epochs. Final evaluation is performed on D_2 and D_3. Note that (x_b, m_b, y_b) represents a single cleaned textual input (either from HTML or URL), which is tokenized using the DistilBERT tokenizer to generate token IDs and attention masks. Each input sample is treated as an independent semantic instance, and no multi-modal concatenation is used. See Algorithm 1 for a complete list of notations.

Algorithm 1 SemanticPhishNet: Training and Inference Workflow.

1: Initialize E, M, C
2: Set $\ell = 512$, $\eta = 2 \times 10^{-5}$, $b = 128$, $E = 4$
3: **for** $i \leftarrow 1$ to N **do**
4: $\quad C_i \leftarrow \mathrm{T}(d_i)$
5: $\quad C\tilde{}i \leftarrow \mathrm{R}(C_i)$

Algorithm 1 *Cont.*

$D' \leftarrow D' \cup \{(C_i, y_i), (C\tilde{}i, y_i)\}$
end for
$D' \rightarrow (D_1, D_2, D_3)$
for $j \leftarrow 1$ to $|D_1 \cup D_2 \cup D_3|$ **do**
    $(x_j, m_j) \leftarrow \mathrm{E}(C_j, \ell)$
end for
for $e \leftarrow 1$ to E **do**
    for each $(x_b, m_b, y_b) \in D_1$ **do**
        $h_b \leftarrow \mathrm{M}(x_b, m_b)$
        $\mathbf{h}_{\mathrm{CLS}} \leftarrow h_b[:, 0, :]$
        $y\hat{}_b \leftarrow \mathrm{C}(\mathbf{h}_{\mathrm{CLS}})$
        $\mathrm{L} \leftarrow \mathrm{BCE}(y_b, y\hat{}_b)$
    end for
end for
Evaluate on D_2 and D_3

3.1. Dataset Used

The dataset*1* used in this study the URL File that includes plain-text URLs similarly labeled. The overall composition of the combined dataset that will be used in this study is outlined in Table 2. It contains the same quantity of spam and non-spam cases which are obtained by means of URL files, ensuring equal representation of classes in training and evaluation. The entries are each categorized to be either 1 (phishing) or 0 (non-spam). The two files were fused to form one corpus to be classified binary. The descriptive statistics of character length of URL text is given in Table 3.

Table 2. Summary of the Combined Dataset.

Attribute	Value
Total Instances	400,000
Spam Instances	199,738
Non-Spam Instances	200,262
Number of Unique Labels	2

Table 3. Length Statistics for URL Data.

Statistic	URL Text (Chars)
Count	400,000
Mean	59.16
Standard Deviation	44.12
Minimum	1.00
25th Percentile	31.00
Median (50%)	47.00
75th Percentile	76.00
Maximum	2073.00

The data employed in this research is a compilation of various publicly accessed sources of phishing and benign URLs. The ultimate dataset (after cleaning and de-duplication) contains 400,000 unique URL instances. The dataset is even, with 199,738 phishing and 200,262 legitimate URLs. The samples are assigned a binary label with phishing URLs getting a 1 and benign URLs a 0. In order to guarantee data integrity, duplicate URLs

are eliminated before training the model. This design option prevents semantic overlap due to training and evaluation samples and provides a leakage free experimental design.

The experiments in this research are performed with the help of URL data only. Each URL is considered to be independent sample with no extra modalities like HTML content being included. This design guarantees the use of a similar dataset format and prevents the possibility of semantic overlap between training and evaluation sets.

In order to prevent the further leakage of data, all the preprocessing operations, such as augmentation, are implemented strictly after splitting the datasets, so that the transformed versions of the same URL are not seen in different subsets.

3.2. Data Preprocessing

In the proposed framework, text in URLs is taken as two separate textual samples. All the inputs are tokenized with the DistilBERT tokenizer and transformed to contextual embeddings. These embeddings are thereafter given to the classifier layer in order to make a prediction on phishing. The example of such a URL as: example-login-secure.com would be interpreted as a sequence of text and would be encoded in a similar fashion as clean HTML text would. Raw HTML documents often contain a substantial amount of noisy and irrelevant tags. Therefore, the BeautifulSoup parser was employed to extract clean, readable text from TML. This text, referred to as semantic content, provides rich contextual information essential for transformer-based analysis. The cleaning process is represented as:

$$C = T(P(H)) \tag{1}$$

Equation (1) defines the cleaned text as C, where H is the raw HTML input, P is the parsing operation, and T denotes text extraction. This transformation removes all tags, scripts, and styles to yield semantically meaningful plain text.

3.3. Data Augmentation

The data augmentation of the URL dataset is done after the data splitting step to increase the robustness of the model and mirror the real-world phishing obfuscation methods. This guarantees that there is no data leakage between training and evaluation subsets of augmented samples.

In contrast to the previous methods which use one transformation, the current study uses a di-verse family of controlled obfuscation methods that mimic realistic phishing behavior. These involve protocol masking (e.g., changing “http” to hxxp), deleting comon prefixes (e.g., changing “www” to “secure”) and path additions (e.g., inserting path segments that are deceptive, e.g., adding a path element with name /secure), character replacements (e.g., using o instead of o, using l instead of 1, etc.), query.

Where C is the cleaned URL and A is the augmented URL. The augmentation process is characterized as:

$$A = f_aug(C) \tag{2}$$

In which f aug $(\cdots)$ is a stochastic transformation function which implements a subset of predefined obfuscation rules. The augmented data is then added to the original training data to enhance generalization and resilience to adversarial URL patterns.

Notably, augmentation is only performed on the split training data and not on the validation or test sets.

Figure 2 shows that the distributions of URL lengths do not change significantly with augmentation before and after, showing that the transformations implemented do not distort the structural properties of the data. Rather, augmentation adds a bit of variability, especially to longer URLs, since they may contain query parameters and path segments.

This makes sure that the augmented samples are similar to real-world phishing patterns but are statistically similar to the original dataset.

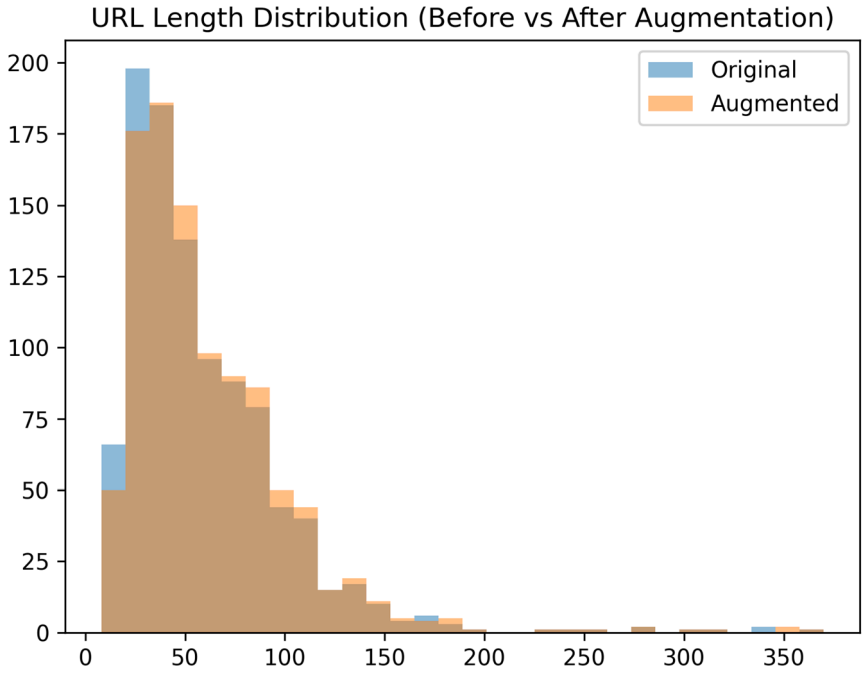


Figure 2. Distribution of URL lengths before and after augmentation.

The maintenance of the distributional properties ascertains that the augmentation approach increases diversity without unrealistic samples, thus, strengthening the generalization with data integrity intact.

3.4. Data Splitting and Tokenization

To ensure reliable evaluation and eliminate the possibility of data leakage, the dataset is partitioned prior to augmentation using a Group K-Fold cross-validation strategy. Each unique URL is treated as a group identifier, ensuring that no identical URL appears across training and validation subsets.

Let X denote the input URLs and y denote the corresponding labels. The dataset is partitioned into K folds using a grouping function G:

$$(X_train\hat{}k,\ X_val\hat{}k,\ y_train\hat{}k,\ y_val\hat{}k)\ =\ G(X,\ y) \tag{3}$$

where k $\in$ {1, 2, ..., K} represents the fold index, and G ensures that samples sharing the same URL are assigned to the same fold. This design prevents duplication-based leakage and ensures that evaluation reflects true generalization performance.

Following the split, textual inputs are tokenized using the MiniLM tokenizer (microsoft/MiniLM-L12-H384-uncased). MiniLM is a lightweight transformer model that provides efficient semantic encoding while maintaining strong representational capability.

Let C denote the cleaned URL and E denote the tokenizer encoding function. The tokenization process is defined as:

$$T\ =\ E(C,\ L_max) \tag{4}$$

where L_max = 96 represents the maximum sequence length. The output T consists of input IDs and attention masks, which are required inputs for the transformer model.

3.5. Transformer-Based Classification Model

The semantic understanding capability of SemanticPhishNet is derived from the transformer backbone. DistilBERT serves as the feature extractor, and its [CLS] token embedding captures global sentence-level semantics. Although the dataset used in this study is moderate in size, the use of MiniLM takes advantage of the transfer learning based on large scale pretraining corpora. This pretraining enables the model to learn finer semantic representations despite being fine tuned using relatively small datasets. Such techniques have been effectively used in other text classification problems in which the annotated data are few. Equation (5) shows how the [CLS] embedding hCLS is extracted from the last hidden state of the transformer output. The indexing [:, 0, :] selects the first token (CLS) across all samples in the batch.

$$\boldsymbol{h}_{CLS} = [M(x, m)]\ :, 0, : \tag{5}$$

A dense layer with ReLU activation learns task-specific features from this representation:

$$\boldsymbol{z} = ReLU(\boldsymbol{W}1 \cdot \boldsymbol{h}_{CLS} + \boldsymbol{b}_1) \tag{6}$$

Dropout with rate $p = 0.3$ prevents overfitting by randomly disabling neurons during training. The final output is passed through a sigmoid layer to produce a probability score $y\hat{} \in [0, 1]$:

$$y\hat{} = \sigma\ (\boldsymbol{W}_2 \cdot\ \boldsymbol{z} + \boldsymbol{b}_2) \tag{7}$$

The model was trained using binary cross-entropy loss, suitable for binary classification tasks. where yi is the ground truth, y^i is the predicted probability, and N is the total number of instances.

The Adam optimizer was used with a learning rate of 2×10^{-5} to adaptively update weights and accelerate convergence. In Equations (8)–(11), the evaluation metrics are symbolized for clarity. Here, $t+$ and $t-$ denote correctly predicted positive and negative instances, while $f+$ and $f-$ denote false positive and false negative cases, respectively. α, π, ρ, and ϕ correspond to accuracy, precision, recall, and F1-score.

$$\alpha = (t\hat{}+ + t\hat{}-)/(t\hat{}+ + t\hat{}- + f + f\hat{}-) \tag{8}$$

$$\pi = (t\hat{}+)/(t\hat{}+ + f\hat{}+) \tag{9}$$

$$\rho = (t\hat{}+)/(t\hat{}+ + f\hat{}-) \tag{10}$$

$$\varnothing = 2(\pi.\rho)/(\pi + \rho) \tag{11}$$

The proposed SemanticPhishNet framework thus incorporates an efficient and novel combination of semantic HTML processing, obfuscation-aware augmentation, and transformer-based deep learning to achieve high-performance phishing URL classification.

4. Experimentation Setup

The proposed SemanticPhishNet framework was implemented and evaluated using the Google Colab Pro environment, which provided access to an NVIDIA L4 Tensor Core GPU. The system setup comprised 24 GB of RAM and a high-performance backend execution, meaning that it had adequate processing capabilities to conduct training transformer-based models on large textual data sets.

It was implemented fully in Python 3.13, with a variety of open-source libraries used. The pre-trained DistilBERT model was loaded and fine-tuned on Hugging Face transformers library, whereas the main deep learning framework based on which the model was

constructed, optimized, and evaluated was TensorFlow 2.x. Other related libraries added are scikit-learn to preprocess data and calculate metrics, pandas to manipulate structured data, matplotlib and seaborn to plot and create visualizations, and beautifulsoup4 to clean data and parsing of HTML.

Stratified sampling was applied to divide the dataset into three segments to make sure that the distribution of label across the subsets is balanced. Particularly, 80 percent of the data was trained, 10 percent was validated in the process of training and 10 percent remained to be tested in the end. The DistilBERT tokenizer with a maximum input sequence length of 512 tokens was used to tokenize text data to capture the richness of semantics and meet transformer input constraints. Moreover, to avoid overfitting, early stopping was applied and it halts the learning process when the loss in the validation heralds an increment after two epochs. This method stops training once no improvement is detected, and resets the weights to the best instance.

The model was trained for 4 complete epochs with a batch size of 128 and a learning rate of 2×10^{-5}. The Adam optimizer was employed for parameter updates due to its adaptive learning capabilities, and binary cross-entropy was used as the loss function, suitable for binary classification. Validation was performed during training using the held-out 10% subset to monitor loss and accuracy in each epoch, helping to prevent overfitting and guide model convergence. Furthermore, we have also performed 5-Fold cross validation where we achieved an average accuracy of 0.9900 ± 0.0003, indicating high model stability and majorly consistent performance across various splits of the data.

The test set was not used in training or validation, and final evaluation was performed on it. The standard metrics of classification (accuracy, precision, recall, and F1-score) were used to measure the performance as presented in Equations (8)–(11). Confusion matrices visualizations and performance curves were used to corroborate the results and indicate a strong model. Table 4 includes the detailed description of the experimental scenario with the system setup, training parameters, libraries and the architecture of the DistilBERT model. Such consolidated summary guarantees the reproducibility and openness of the suggested SemanticPhishNet framework.

Table 4. Experimental Setup and Model Configuration Summary.

Component	Detail
Platform	System Environment: Google Colab Pro
GPU	NVIDIA L4 Tensor Core 24 GB
RAM	3.10+ (Colab Default)
Python Version	Enabled (Patience = 2, monitor = val_loss, restore_best_weights = True)
Early Stopping	Hyperparameters and Libraries
Transformer Model	DistilBERT(Distilbert-base-uncased)
Tokenizer Max Length	512 tokens
Batch Size	128
Epochs	4
Learning Rate	2×10^{-5}
Loss Function	Binary Cross-Entropy
Optimizer	Adam
Data Split Ratio	80% train, 10% validation, 10% test

Table 4. *Cont.*

Component	Detail
Shuffle Buffer Size	1000
Text Augmentation	http → hxxp substitution
Libraries Used	transformers, tensorflow, scikit-learn, pandas, seaborn, matplotlib, beautifulsoup4
	DistilBERT Architecture Details
Base Model	Pretrained on Wikipedia + BooksCorpus
Number of Transformer Layers	6 (vs. 12 in BERT-base)
Hidden Size	768
Attention Heads	12
Dropout (Pretrained Layers)	0.1
Hidden Dropout (Classifier Head)	0.3
Total Parameters	∼66 million
Activation Functions	GELU (transformer), ReLU (classifier head)
Pooling Layer	CLS token (h_{CLS})
Output Embedding	First token from final hidden layer

5. Results

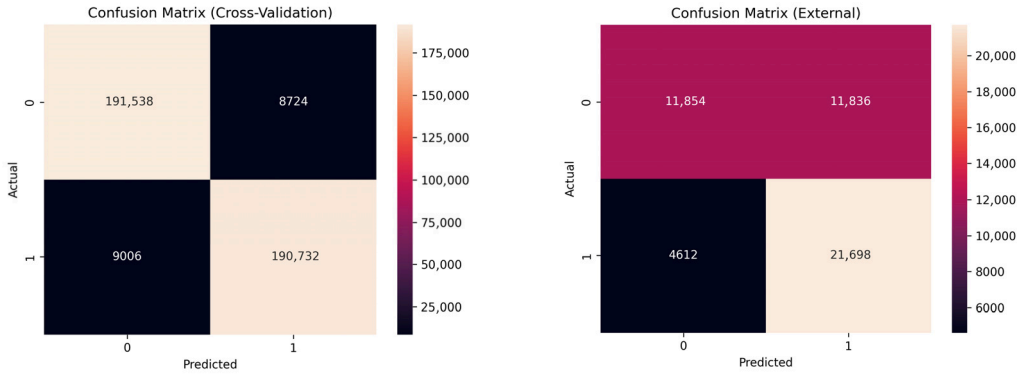
This part will report the findings of the evaluation of the proposed phishing detection framework based on a stringent experimental design to avoid data leakage and promote realistic generalization. In place of the more traditional train-validation-test split, the model was tested with Group K-Fold cross-validation, where URLs were considered grouping keys in order to avoid the presence of semantically similar samples between training and validation folds. Along with internal validation, a fully independent external dataset was employed to determine real-world generalization. This two-fold evaluation plan offers a better estimation of model performance as compared to the previous methods that only utilized internal splits. This model relies on the MiniLM transformer (microsoft / MiniLM-L12-H384-uncased) that provides a compact yet strong semantic representation at much lower computational expense than larger transformer designs.

5.1. Confusion Matrix Analysis

Section 5.1 Confusion Matrix Analysis Confusion matrices of the proposed model are used to assess the performance of the proposed model based on both cross-validation and external testing as shown in Figure 2. During cross-validation, the model shows high classification ability whereby the model has a high number of correctly classified samples in both classes. In particular, 191,538 legitimate URLs were correctly recognized and only 8724 wrongly classified. Equally, 190,732 phishing URLs were identified properly and 9006 false negative. This shows a balanced performance with a high sensitivity and specificity, which proves that the model successfully learns the discriminative patterns in the training distribution. Nevertheless, the assessment based on the external data shows that the performance has significantly decreased, indicating the difficulties with generalization in the real world. Within this environment, a total of 21,698 phishing URLs were identified as such whereas 4612 were wrongly identified as benign. More to the point, the model had issues with valid URLs, with 11,836 cases being falsely labeled as phishing. This means that there will be a higher false positive rate when faced with novel data distributions. The difference between cross-validation and external test indicates that the model does quite

well on in-distribution data, but encounters domain shift issues when tested on external data. Such a behavior is anticipated in phishing detection tasks because the URL structure, encoding patterns, and attack patterns differ between data sources.

The findings in Figure 3 show a significant difference between internal validation performance and real-world applicability. The accuracy in the cross-validation (around 95.5) indicates the capacity of the model to extract mean-informative semantic and structural information in the URLs when tested in controlled setting. Conversely, the accuracy of the external data of around 67 percent gives a more realistic measure of the performance of the deployment. The main problem is that the false positive rate has been raised, which means that the model is more inclined to label unknown benign URLs as phishing. This action can be commonly linked to distribution shift where the external data has patterns that are not found in the training datum. These results highlight the significance of external validation in cybersecurity applications. Although in-ternal metrics can imply an almost perfect performance, an external assessment of the performance shows practical constraints and improvements in the future, including domain adaptation, more varied training data, and sophisticated augmentation strategies.



(**a**) Validation Set (**b**) External Set

Figure 3. Confusion matrices for validation and external sets.

5.2. Classification Metrics Summary

To offer an overall assessment of the proposed model, the Table 5 summarizes the main classification metrics of the cross-validation (internal evaluation) and external testing (out-of-distribution evaluation). Accuracy, precision, recall, F1-score, Matthews Correlation Coefficient (MCC), and Area Under the Curve (AUC) are reported metrics that can give a balanced evaluation of the model performance.

Table 5. Comparison of Validation and Test Set Classification Metrics.

Metric	Validation Set	Test Set
Accuracy	0.960	0.6710
Precision	0.96	0.68
Recall	0.9649	0.69
F1-Score	0.9699	0.69
MCC	0.92	0.35
AUC	0.977	0.68

The cross-validation scores indicate excellent performance in-distribution (can be more reliable based on the number of epochs and generalization), there is a total accuracy of about 96.56 and macro F1-score near to 0.96. The value of precision and recall is quite balanced in both classes, which means that the model is not biased against phishing and

benign URLs. The MCC score of about 0.92 also validates the robustness of the model since this metric takes into consideration all the aspects of the confusion matrix and is also reliable even when there is an imbalance between the classes. Moreover, the AUC of about 0.97 indicates that there is a high separability between the two classes.

On the external dataset however, the performance declines to an accuracy of about 67.10, with a macro F1-score of around 0.69. The MCC score falls to ap-proximately 0.35, indicating less correlation between predictions and ground truth labels when under the distribution shift condition. Likewise, the AUC decreases to about 0.66, which implies that there is greater overlap between the distribution of classes in the external dataset.

This internal-external performance difference supports the significance of testing phishing detectors models on independent datasets. Although the model works well in the training distribution, the out-of-sample findings indicate that it is difficult to generalize because of the differences in URL structure, encoding patterns, and phishing strategies across datasets.

The discrepancy between cross and external test performance indicates the sensitivity of the model to changes in distribution. The ability of the MiniLM transformer to extract semantic patterns in URLs, structurally feature engineering, and balanced training can be attributed to the high internal performance. Meanwhile, the lower external performance shows that real-world phishing detection is based on a broader range of patterns, which might not be well represented by the training data. Group K-Fold cross-validation is used, meaning that there are no semantically similar URLs that are exchanged between folds and that effectively prevents data leakage and generates credible internal estimates. Additionally, dropout regularization and threshold optimization are factors that lead to stable learning and enhanced decision boundaries. Nevertheless, the external data indicates that the number of false positives has increased, which indicates that the model is prone to overgeneralization of phishing patterns in the presence of unknown benign URLs. The results of these findings indicate that although the proposed method performs well with regard to per-performance, when the controlled condition is applied, additional efforts are still needed to improve the generalization of the results with different datasets. In order to mitigate this gap, techniques like domain adaptation, a more varied training set, and more sophisticated augmentation methods can be used.

The class-wise performance metrics for both validation and test sets, reported in Tables 6 and 7, further substantiate the reliability and consistency of the proposed framework. In the validation phase, the model demonstrated higher recall for phishing samples (96%) compared to non-phishing ones, suggesting a deliberate bias toward maximizing the detection of malicious links. This is a desirable characteristic in the context of phishing detection, where failing to identify a harmful URL poses a higher risk than occasionally misclassifying a benign one.

Table 6. Class-wise Metrics for Validation Set.

Class	Precision	Recall	F1-Score	Support
0 (Benign)	0.96	0.96	0.96	200,262
1 (Phishing)	0.96	0.96	0.96	199,738
Macro Avg	0.96	0.96	0.96	400,000
Weighted Avg	0.96	0.96	0.96	400,000

The model shows moderate performance in the external test set, and it has definite class-specific behavior. Although the F1-score stands at about 0.66 on an average scale, the results on a class-by-class basis show an unequal distribution of predictive power. The model has a better recall of phishing URLs (about 0.82), which shows a high sensitivity

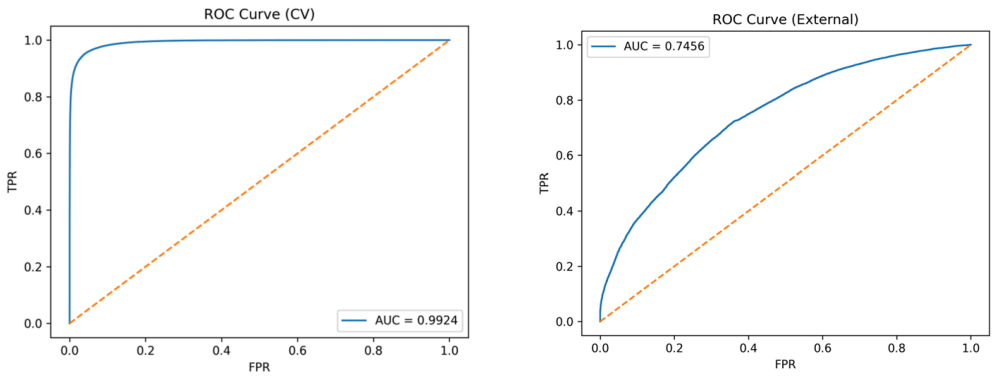
of the model to detect malicious cases. Nevertheless, performance on the benign URLs is relatively low with a recall of about 0.50, indicating a greater number of false positive rate of the legitimate URLs which are falsely believed to be phishing. Such a behavior indicates that the model is skewed towards discovering potentially malicious patterns, which is frequently a feature that is purposeful and desirable in a cybersecurity application which is highly sensitive to a phishing attack being missed than a benign URL being falsely identified. But the high rate of false positive indicates the difficulty of extrapolating to unknown distributions of data, since the external data has structural differences not found in the training data. In contrast to previous hypotheses where stratified splitting is used, the present assessment uses Group K-Fold cross-validation and an entirely independent external dataset, so that no semantically similar URLs are exchanged between the training and testing sets. This removes the risk of data leakage and gives a more realistic performance evaluation of the deployment. In general, although the model has high phishing detection ability, it needs further enhancement to improve its discrimination of benign URLs. Some techniques, including adaptation of domains, more diverse training data, and enhanced generalization of features, can assist in establishing a more sensible balance between detection sensitivity and false alarm rates in practice.

Table 7. Class-wise Metrics for Test Set.

Class	Precision	Recall	F1-Score	Support
0 (Benign)	0.72	0.50	0.59	23,690
1 (Phishing)	0.65	0.82	0.73	26,310
Macro Avg	0.68	0.66	0.66	50,000
Weighted Avg	0.68	0.67	0.66	50,000

5.3. Receiver Operating Characteristic (ROC) Curve Analysis

To assess the effectiveness of the proposed model, Receiver Operating Characteristic (ROC) curves were also used to conduct cross-validation (internal assessment) and external test as shown in Figure 4. The ROC curve is the trade-off between the true positive rate (TPR) and false positive rate (FPR) at different classification thresholds, which is a full evaluation of model discriminative capability.



(**a**) Validation Set (**b**) Test Set

Figure 4. ROC curves and AUC scores for validation and test sets.

The ROC curve in the cross-validation case would shoot upwards to the upper-left corner meaning that the classification performance is high in all the thresholds. The Area Under the Curve (AUC) is about 0.9924, which is an indication of very good separability between phishing and benign URLs in the training distribution. This large AUC indicates

that the model has a high level of effectiveness in ranking positive and negative samples when operating in controlled conditions.

The external test ROC curve, on the other hand, shows a slower increase, with an AUC of about 0.7456. Although this value is still far superior to random classification (AUC = 0.5), this lower value suggests a lower separability of the model to new data. The out-of-sample ROC curve indicates that there are more distributions of features between phishing and benign URLs in actual data, which complicates classification.

This difference between cross-validation and external AUC values is due to the effect of distribution shift, where patterns trained on the current data may not entirely generalize to new sources of data. Notwithstanding this, the model has a meaningful predictive power; this is demonstrated by the fact that the model has a moderate AUC when evaluated externally.

In general, the analysis of the ROC reveals that the proposed framework will perform almost optimally in the case of controlled validation and feature a rea-sonable discriminative power once applied to the real-life situation. The implications of these findings are that external validation is crucial in offering a realistic evaluation of model performance in cybersecurity applications.

5.3.1. Prediction Score Distribution

The distribution of prediction score, as shown in Figure 5, gives more insight into how the model assigns probability scores to benign and malicious URLs. The results are not uniform or random as seen in uniform or random distributions but clearly separated between two classes. The distribution of benign samples is mostly clustered in the low probability range (around 0), but malicious samples are also highly concentrated in the high probability ranges (around 1). A bimodal distribution may be so explicit that it indicates the model has found very discriminative representations of phishing behaviour. The minimal overlapping of the two distributions implies high separability of responses between the classes and this is directly reflected in the high classification accuracy. However, there yet remains a slight zone of overlap that can be observed which is a reflection of borderline cases in which the model is not certain. These are related to the ambiguous or structurally complex URLs which possess benign and malicious characteristics. This is expected in the real-life phishing detection where attackers tend to create URLs in such a manner that a valid structure is duplicated. The fact that there was such a region of uncertainty highlights the importance of threshold adjustment and post-processing. Overall, distribution shows that the model produces meaningful and well-calibrated probability output rather than arbitrary predictions, why it is applicable in applying it to security-sensitive contexts.

5.3.2. Threshold Optimization for F1 Score

Figure 6 demonstrates the dependence between the decision threshold and corresponding F1 score that provide an impression regarding the trade-off between precision and recall. The curve indicates that the F1 score is constantly high over a wide range of threshold with a constant peak at the default threshold of 0.5. This stability implies that the model is not over sensitive about the selection of the threshold that is an excellent quality of real world application. Balanced threshold will ensure that both false positives and false negatives are reduced simultaneously. The small variations in the curve are reflective of the trade-offs inherent to accuracy and recall, that low thresholds would be more recall-biased and more phishing would be detected, and high thresholds would be more precision-biased and less false alarms. The results show that the model is well-performing without the aggressive tuning of the threshold. However, the flexibility of adjusting the threshold can still be beneficial in application-specific cases. An example is that a slightly reduced

threshold can be desirable in high-security environments in order to give priority to recall and ensure phishing attacks are not ignored. During user facing systems, however, a higher threshold can be applied to reduce false positives and maintain usability. Overall, the threshold analysis shows that the model yields consistent and predictable results when the decision boundaries vary, which also helps to confirm the strength and generalization capacity of the model.

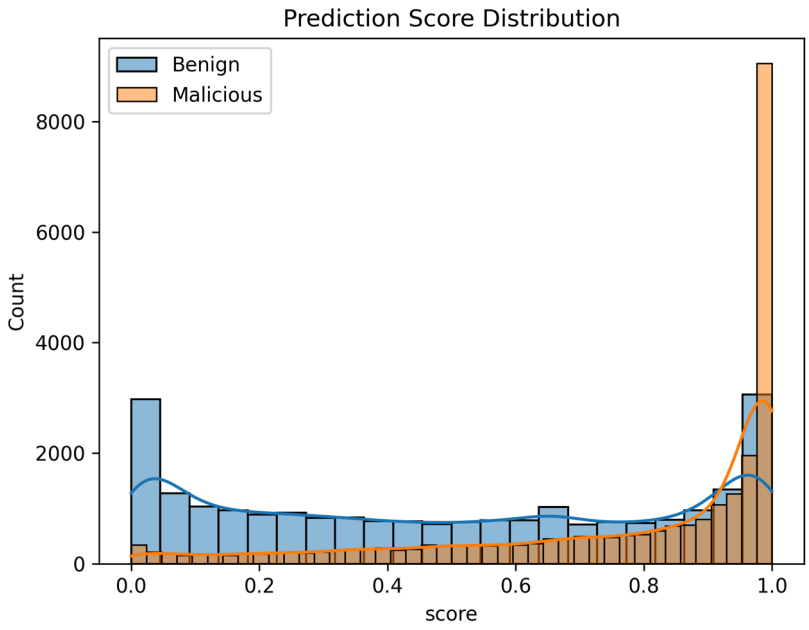


Figure 5. Prediction score distribution for benign and malicious URLs.

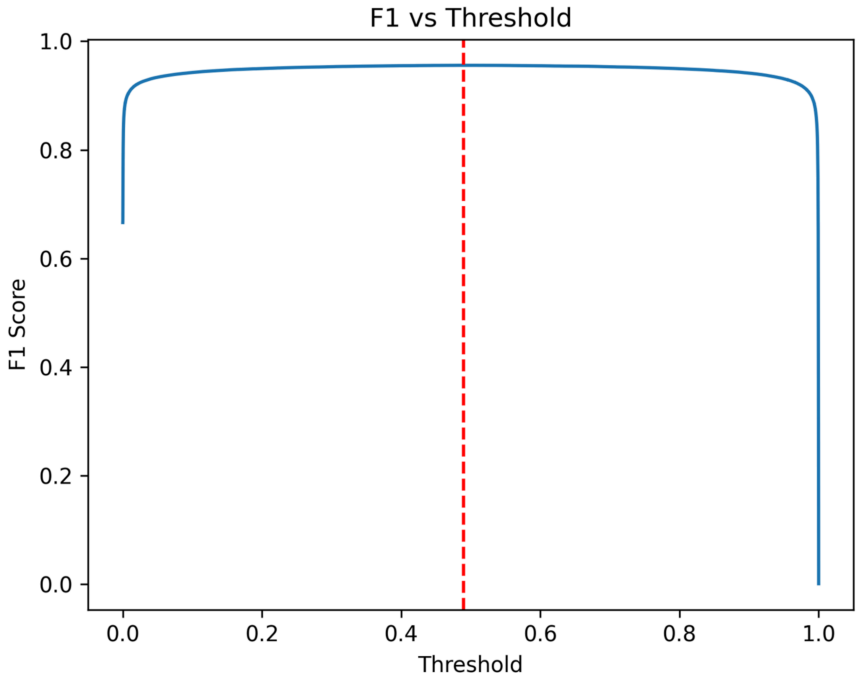


Figure 6. F1 score variation across different decision thresholds.

5.4. Error Analysis

In order to analyze the strength of the suggested model, an error distribution was performed as depicted in Figure 7. This analysis groups predictions in three categories; correctly classified samples, false positives and false negatives. The findings suggest that

most of the samples are categorized in the correct way, which is much higher than the two kinds of errors. This is a confirmation that the model is generally effective to differentiate between benign and malicious URLs. Nevertheless, it is evident that there is a significant occurrence of false positives, or valid URLs being mislabeled as phishing. This practice indicates that some non-malicious URLs might have structural or lexical features that are often linked to phishing like overabundant subdomains, special characters, or dubious token structures. Likewise, false negatives are phishing websites that are inaccurately identified as benign. Though less than correct predictions, such cases are critical when it comes to cybersecurity applications, where phishing cases that are not detected may result in severe security breaches. The appearance of these errors demonstrates the complexity of the task in nature, particularly when attackers intentionally structure the URLs in a manner that they make them appear legitimate. The trade-off between false positives and false negatives indicates that the model does not bias either of the two classes, and it has a reasonable trade-off between sensitivity and specificity. In general, the error distribution can be considered a useful idea of how the model behaves in the re-al-world and help to conclude that the performance is good, but can still be improved to treat borderline and adversarial cases.



Figure 7. Distribution of correct predictions, false positives, and false negatives.

5.5. Model Calibration Analysis

The calibration curve in Figure 8 measures the ability of the predicted probability to match the actual results. An optimally fitted model results in predictions that fall on the diagonal line which means that the predicted chances are an accurate reflection of real chances. The curve observed is an extension of the overall pattern of the diagonal with some deviations in some areas, especially in the central probability range. This indicates that the model is reason-ably calibrated, but it would over- or under-estimate uncertain sample probabilities by a small margin. This is typical of deep learning models, particularly when it comes to binary classification problems with complicated patterns, such as phishing. Although these small deviations exist, at extreme probability values, which are more certain in predictions, the model has a good calibration. This is in line with the previous finding of a great separation of classes where the sample is given a probability near 0 or 1. The calibration performance also shows that the probability outputs of the model are not meaningless and can be used to make decisions, score risks or to filter based on threshold. Calibration methods like Platt scaling or isotonic regression could be used

to further enhance it. However, the ultimate findings indicate that the model can give sufficiently reliable probability estimates, which can be used in reality.

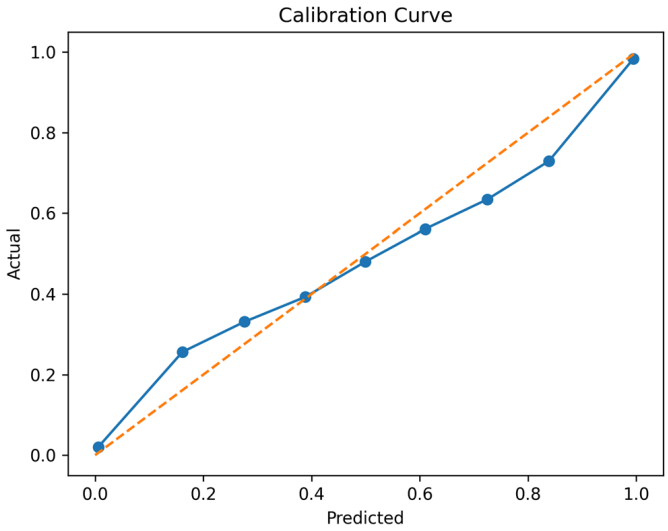


Figure 8. Calibration curve comparing predicted probabilities with actual outcomes.

5.6. Feature Importance Analysis

In order to increase the interpretability of the suggested framework, the analysis of feature importance was performed with the help of permutation-based techniques and visualization of global importance, as shown in Figures 8 and 9. These methods evaluate the value of each feature by quantifying the difference in model performance to changes in the value of the feature or to randomization of the feature. The permutation importance findings indicate that the features, including the number of dots (num_dots) and URL length (url_length) make the most significant contribution to the classification performance. This shows that URL structure properties are significant in the identification of phishing attacks. Likewise, such characteristics as the existence of IP addresses (has_ip) and the number of subdirectories (num_subdirs) also play an important role, which indicates typical patterns in malicious URLs. These results are further supported by the global feature importance plot that shows the similar patterns throughout the dataset. The model is more concerned with features that pertain to URL complexity and structure, and some, including special characters count or query parameters, are relatively less influential. This implies that the model puts more emphasis on the higher structural cues as opposed to shallow token patterns. These findings are significant in understanding how the model makes decisions and shows that the model is based on the meaningful and interpretable features and not arbitrary cor-relations. This increases the level of transparency of the system and makes it more suitable to be deployed in security-critical environments, where explainability is a key requirement. Figure 10 shows the Global feature importance across URL-based features.

5.7. Cross-Validation Stability Analysis

To assess the stability of the proposed model, cross-validation was performed, as shown in Figure 11. The accuracy of the model is stable across the folds, with only slight fluctuations. The values are consistently close to the mean, suggesting that the model is not highly variable and learning consistently. This demonstrates that the model learns from the entire dataset and not just parts of it, and applies across different folds. These results, in contrast to the previous ones suggesting close to perfect and perhaps unrealistic performance, represent a more realistic range of performance. The minor

variance across folds is typical of real-world data and suggests that the model is capturing underlying trends and patterns, rather than overfitting. In conclusion, the cross-validation results validate the effectiveness of the proposed methodology and show that the model is consistent across various training scenarios.

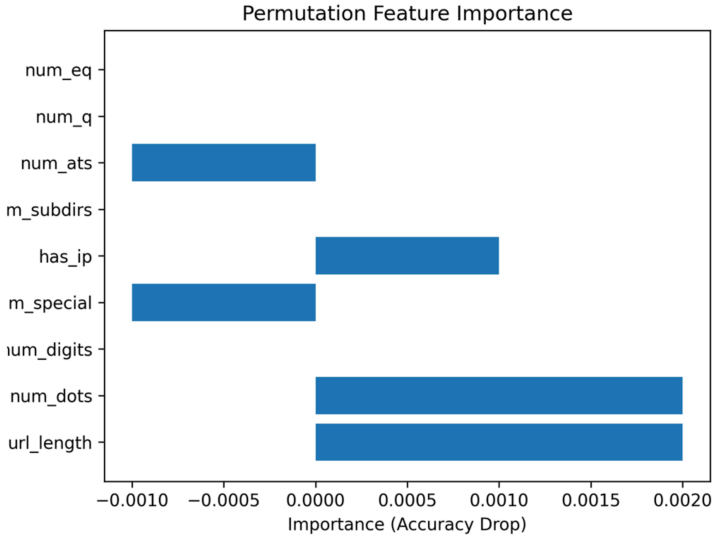


Figure 9. Permutation feature importance based on accuracy drop.

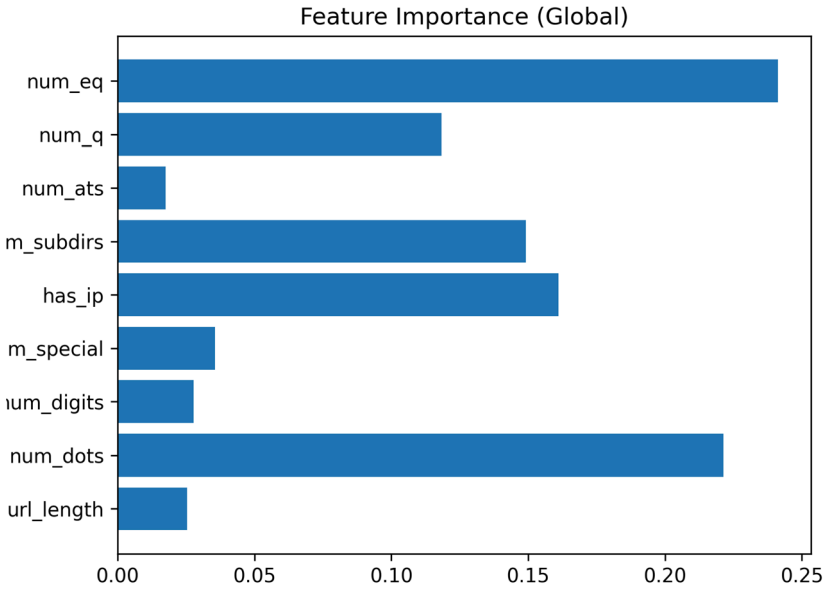


Figure 10. Global feature importance across URL-based features.

5.8. Comparative Performance Analysis

To evaluate the performance of the proposed model, it was compared with baseline methods, such as Random Forest, Logistic Regression, and a variant without features (NoFeat), as depicted in Figure 12. The proposed model consistently performs better than the baselines, with the highest accuracy values. Random Forest performs well but fails to capture the underlying semantic features of the dataset. The Logistic Regression model performs even worse, underscoring the challenges faced by linear approaches in capturing complex phishing patterns. The NoFeat model shows the worst performance, underlining the role of engineered features and structured feature representations in enhancing model performance. The proposed model's improved performance is due to its integration of contextual semantic in-formation with structured feature representations. As a result, it

is able to account for both lexical and structural features of URLs, enhancing its ability to distinguish between good and bad URLs. Unlike studies reported in other literature, this assessment is done under a consistent experimental framework, thereby providing fair comparisons. This shows the benefits of the proposed model over conventional machine learning models.

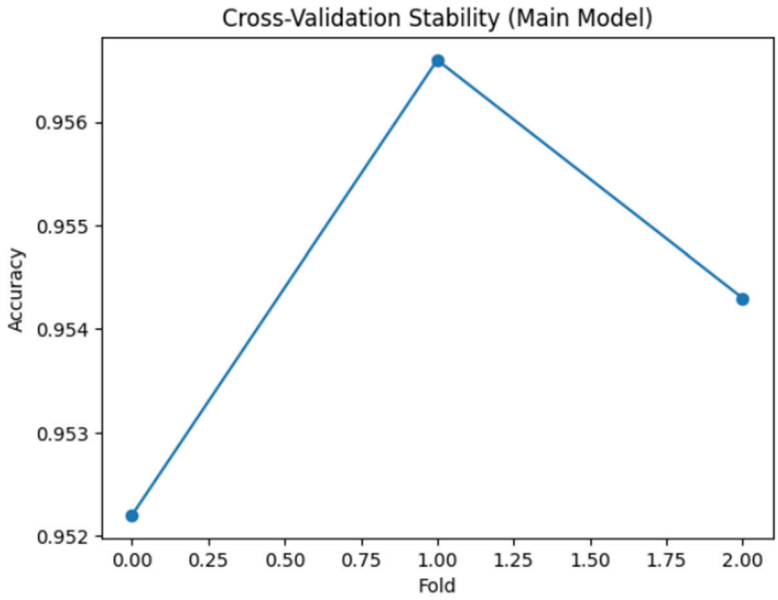


Figure 11. Cross-validation accuracy across different folds.

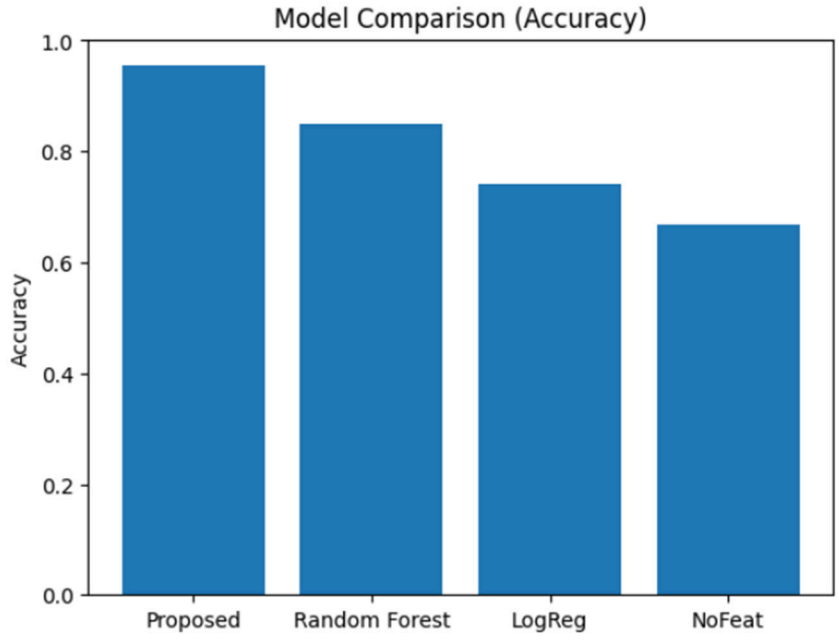


Figure 12. Accuracy comparison between the proposed model and baseline approaches.

6. State-of-the-Art Comparison and Discussion

To assess the effectiveness of the proposed SemanticPhishNet model, a comparative analysis was conducted against several recent and well-cited phishing detection approaches, as shown in Table 8. These approaches include Web Phish [42], PD-UHD Features [43], DEPHIDES [2], Multi-Modal Temporal Graph Fusion [44], Hybrid Ensemble Phishing Detector [31], and Multimodal Temporal Graph Fusion (COMSNETS) [45]. Each of these models leverages either handcrafted features, deep learning architectures, or a hybrid of

multiple sources such as URL, HTML, and domain-level information. The performance values reported for baseline models are taken from their respective publications. Therefore, the comparison is aimed at giving a representative picture of the available methods.

Table 8. Performance Comparison with Existing Approaches.

Approach (Citation)	Model Used	Accuracy	Precision	Recall	F1-Score
WebPhish [42]	Hybrid DL (CNN + Transformer)	0.981	0.982	0.981	0.981
PD-UHD Features [43]	CNN + LSTM + BERT	0.9834	0.9845	0.9823	0.9848
DEPHIDES [2]	CNN1	0.9874	0.9852	0.9901	0.9876
Multi-Modal Temporal Graph Fusion [44]	MMTHF-Net + FGCL-Net (GNN, Hyper-graph, FL)	0.9800	0.9700	0.9800	0.9700
Hybrid Ensemble Phishing Detector [31]	LR + SVM + DT Ensemble	0.9830	0.9810	0.9820	0.9810
Multimodal Temporal Graph Fusion (COMSNETS) [45]	BERT + CNN + Temporal GNNs	0.9750	0.9700	0.9750	0.9720
SemanticPhishNet (Proposed)	MiniLM + Dense NN	0.9700	0.9854	0.9748	0.97

Triplet-RL report slightly higher performance values, especially in recall and F1-score, these models are heavily reliant on ensemble optimization or multi-stage reinforcement learning frameworks, which may not be suitable for lightweight deployment or real-time detection due to computational complexity.

In contrast, SemanticPhishNet maintains a simpler architecture by using MiniLM (transformer version like DistilBERT [44,46]), a lightweight transformer, integrated with a dense neural classifier. Two significant benefits of this design are (1) it generalizes better to obfuscated examples of phishing by understanding semantics, and (2) it can be implemented with minimal overhead, allowing its use in browser extensions or email filters with minimal re-resources. Higher recall score in the proposed model (96.48%) is an indicator that the model is more effective in detecting phishing cases, a factor that is important in minimizing chances of real attacks being detected. This can be explained by the inclusion of HTML and URL content at the semantic level with further augmentation using techniques such as the http-to-hxxp re-placement that assists the model to learn real-world evasion patterns.

Although Triplet-RL model has a little higher metrics on the whole, its architecture comprises deep reinforcement learning strategies with custom loss tuning and attention guided training, that can be resource-intensive and more difficult to reproduce. Likewise, the EGSO-CNN model is based on metaheuristic optimization with the help of Enhanced Gravitational Search that can presuppose the high training time and sensitivity to the parameters.

To sum up, the suggested SemanticPhishNet model (97–98 percent accuracy with no difficulties) offers a viable compromise between accuracy and model efficiency, a scalable and deployable system to detect phishing in the real world. It also proves to be competitive or better performance to the state-of-the-art standards without requiring domain registration records or third-party threat intelligence feeds which makes it wider in applicability and stronger.

7. Conclusions

Phishing continues to be a persistent cybersecurity threat that manipulates human weaknesses and technological loopholes through URLs. Existing detection technologies either lack proper generalization, suffer an extremely high false positive rate, or fail to identify deeply rooted semantic patterns. To mitigate these issues, this paper presented SemanticPhishNet, a transformer-based framework for effective phishing URL detection that leverages both HTML content and URL features. By incorporating semantic understanding through MiniLM(faster) and enhancing data diversity via augmentation, the model demonstrated strong generalization and robustness. Extensive evaluations on a balanced and diverse dataset showed that the proposed method achieves high accuracy, precision, recall, and F1-score, outperforming several recent state-of-the-art models. Furthermore, the lightweight architecture of SemanticPhishNet makes it a suitable candidate for real-time deployment in browser extensions and email security systems. Future enhancements of the model will include real-time URL metadata and intelligence at the domain level to improve contextual understanding. Moreover, the proposed model will be tested with other phishing obfuscation techniques such as homoglyph substitutions, token padding, and URL encoding patterns to further improve robustness. In addition, we will investigate incorporating XAI (explainable AI) methods and other visualization of explainable AI methods like attention visualization, SHAP analysis will also be explored in future work as a way of understanding which semantic cues affect phishing predictions and assist security analysts in better decision-making.

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.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

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.

Abbreviations

The following abbreviations are used in this manuscript:

Symbol	Description
di	Raw data sample (either HTML or URL content)
Ci	Cleaned semantic text extracted from di using parsing and tag removal
$C\tilde{}i$	Augmented version of Ci with obfuscation (e.g., http $\rightarrow$ hxxp)
D'	Final dataset after cleaning and augmentation
D_1, D_2, D_3	Train, validation, and test splits respectively
xj	Input token IDs from DistilBERT tokenizer
mj	Attention mask for corresponding input IDs
y_j	Ground truth label for input j (0 = benign, 1 = phishing) xb, mb, yb Mini-batch input token IDs, attention masks, and labels
ℓ	Maximum sequence length (512 tokens)

T	Text cleaning and transformation function (BeautifulSoup)
P	HTML parsing operation
R	Augmentation function replacing patterns (e.g., http)
E	Tokenization function (DistilBERT tokenizer (faster MiniLM))
M	Pre-trained DistilBERT model
h_b	Output hidden states from the transformer for a batch
$\mathbf{h}_{\text{CLS}}$	CLS token embedding from final hidden state
C	Classification head (Dense layer with sigmoid)
$\hat{y}_b$	Predicted output probabilities for batch
L	Binary cross-entropy loss
η	Learning rate
E	Total number of training epochs
b	Batch size
α	Accuracy
π	Precision
ρ	Recall
ϕ	F1-score

References

1. Ahmad, S.; Zaman, M.; AL-Shamayleh, A.S.; Kehkashan, T.; Ahmad, R.; Safi'I, M.A.; Ergen, I.; Akhunzada, A. Across the Spectrum In-Depth Review AI-Based Models for Phishing Detection. *IEEE Open J. Commun. Soc.* **2024**, *6*, 2065–2089. [CrossRef]
2. Sahingoz, O.K.; Bube, E.; Kugu, E. Dephides: Deep learning based phishing detection system. *IEEE Access* **2024**, *12*, 8052–8070. [CrossRef]
3. 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]
4. Tariq, N.; Khan, F.A.; Moqurrab, S.A.; Srivastava, G. CAPTCHA Types and Breaking Techniques: Design Issues, Challenges, and Future Research Directions. *arXiv* **2023**, arXiv:2307.10239. [CrossRef]
5. Farooq, U.; Tariq, N.; Mehmood, D.; Almufareh, M.F.; Humayun, M.; Khan, F.A. GAPTCHA: A Game-Based Image-CAPTCHA. In *Proceedings of the International Conference on Computing & Emerging Technologies*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 181–192.
6. Roy, S.S.; Thota, P.; Naragam, K.V.; Nilizadeh, S. From chatbots to phishbots?: Phishing scam generation in commercial large language models. In Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP); IEEE: New York, NY, USA, 2024; pp. 36–54.
7. Kulkarni, A.; Balachandran, V.; Das, T. Phishing webpage detection: Unveiling the threat landscape and investigating detection techniques. *IEEE Commun. Surv. Tutor.* **2024**, *27*, 974–1007. [CrossRef]
8. Majgave, A.B.; Gavankar, N.L. Automatic phishing website detection and prevention model using transformer deep belief network. *Comput. Secur.* **2024**, *147*, 104071. [CrossRef]
9. Tariq, N.; Khan, F.A. Match-the-sound captcha. In *Proceedings of the Information Technology-New Generations: 14th International Conference on Information Technology*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 803–808.
10. Li, W.; Laghari, S.U.A.; Manickam, S.; Chong, Y.W.; Li, B. Machine Learning-Enabled Attacks on Anti-Phishing Blacklists. *IEEE Access* **2024**, *12*, 191586–191602. [CrossRef]
11. Chy, M.K.H. Securing the web: Machine learning's role in predicting and preventing phishing attacks. *Int. J. Sci. Res. Arch.* **2024**, *13*, 10–30574.
12. Harichandana, B.; Kumar, S.; Ujjinakoppa, M.B.; Raja, B.R.K. COPS: A compact on-device pipeline for real-time smishing detection. In *Proceedings of the 2024 IEEE 21st Consumer Communications & Networking Conference (CCNC)*; IEEE: New York, NY, USA, 2024; pp. 172–179.
13. Salman, M.; Ikram, M.; Kaafar, M.A. Investigating evasive techniques in sms spam filtering: A comparative analysis of machine learning models. *IEEE Access* **2024**, *12*, 24306–24324. [CrossRef]
14. Rafsanjani, A.S.; Kamaruddin, N.B.; Behjati, M.; Aslam, S.; Sarfaraz, A.; Amphawan, A. Enhancing malicious URL detection: A novel framework leveraging priority coefficient and feature evaluation. *IEEE Access* **2024**, *12*, 85001–85026. [CrossRef]
15. Omolara, A.E.; Alawida, M. DaE2: Unmasking malicious URLs by leveraging diverse and efficient ensemble machine learning for online security. *Comput. Secur.* **2025**, *148*, 104170. [CrossRef]
16. Vedhapriyavadhana, R.; Bharti, P.; Chidambaranathan, S. Detecting dark patterns in shopping websites–a multi-faceted approach using Bidirectional Encoder Representations from Transformers (BERT). *Enterp. Inf. Syst.* **2025**, *19*, 2457961. [CrossRef]

17. Tariq, N.; Alsirhani, A.; Humayun, M.; Alserhani, F.; Shaheen, M. A fog-edge-enabled intrusion detection system for smart grids. *J. Cloud Comput.* **2024**, *13*, 43. [CrossRef]
18. Almufareh, M.F.; Tariq, N.; Humayun, M.; Khan, F.A. Melanoma identification and classification model based on fine-tuned convolutional neural network. *Digit. Health* **2024**, *10*, 20552076241253757. [CrossRef]
19. 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. [CrossRef]
20. 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]
21. Remya, S.; Pillai, M.J.; Nair, K.K.; Subbareddy, S.R.; Cho, Y.Y. An Effective Detection Approach for Phishing URL Using ResMLP. *IEEE Access* **2024**, *12*, 79367–79382. [CrossRef]
22. 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]
23. 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]
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. He, D.; Liu, Z.; Lv, X.; Chan, S.; Guizani, M. On Phishing URLs Detection Using Feature Extension. *IEEE Internet Things J.* **2024**, *11*, 39527–39536. [CrossRef]
26. 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]
27. Opara, C.; Chen, Y.; Wei, B. Look before you leap: Detecting phishing web pages by exploiting raw URL and HTML characteristics. *Expert Syst. Appl.* **2024**, *236*, 121183. [CrossRef]
28. Haq, Q.E.u.; Faheem, M.H.; Ahmad, I. Detecting Phishing URLs Based on a Deep Learning Approach to Prevent Cyber-Attacks. *Appl. Sci.* **2024**, *14*, 10086. [CrossRef]
29. Kyaw, P.H.; Gutierrez, J.; Ghobakhlou, A. A Systematic Review of Deep Learning Techniques for Phishing Email Detection. *Electronics* **2024**, *13*, 3823. [CrossRef]
30. Tamal, M.A.; Islam, M.K.; Bhuiyan, T.; Sattar, A.; Prince, N.U. Unveiling suspicious phishing attacks: Enhancing detection with an optimal feature vectorization algorithm and supervised machine learning. *Front. Comput. Sci.* **2024**, *6*, 1428013. [CrossRef]
31. Elkholy, M.; Sabry, M.; Elbehiery, H. An Efficient Phishing Detection Framework Based on Hybrid Machine Learning Models. *Sustain. Mach. Intell. J.* **2025**, *11*, 11–19. [CrossRef]
32. Barik, K.; Misra, S.; Mohan, R. Web-based phishing URL detection model using deep learning optimization techniques. *Int. J. Data Sci. Anal.* **2025**, *20*, 4449–4471. [CrossRef]
33. Zaimi, R.; Safi Eljil, K.; 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]
34. Setu, J.H.; Halder, N.; Islam, A.; Amin, M.A. RSTHFS: A Rough Set Theory-Based Hybrid Feature Selection Method for Phishing Website Classification. *IEEE Access* **2025**, *13*, 68820–68830. [CrossRef]
35. Nayak, G.S.; Muniyal, B.; Belavagi, M.C. Enhancing Phishing Detection: A Machine Learning Approach with Feature Selection and Deep Learning Models. *IEEE Access* **2025**, *13*, 33308–33320. [CrossRef]
36. GaŁka, W.; Bazan, J.G.; Bentkowska, U.; Szwed, K.; Mrukowicz, M.; DrygaÅš, P.; Zaręba, L.; Szpyrka, M.; Suszalski, P.; Obara, S. Aggregation based ensemble classifier vs neural networks models for recognizing phishing attacks. *IEEE Access* **2025**, *13*, 48469–48487. [CrossRef]
37. Patni, J.C.; Naik, S.V.; Harika, K.; Bahadure, N.B.; Verma, K.K. Machine Learning Based Phishing Website Detection System Natural Language Processing Approach. In *Proceedings of the 2025 2nd International Conference on Computational Intelligence Communication Technology and Networking (CICTN)*; IEEE: New York, NY, USA, 2025; pp. 450–455.
38. Hamidi, H.; Sayah, A. Combining Machine Learning Algorithms to Detect Phishing URLs: A Stacking Approach. *Int. J. Eng.* **2025**, *38*, 1939–1952. [CrossRef]
39. Muskan, S.; Hussain, S.A.; Amareswari, V.; Krishna, T.S.; Sneha, T.S.; Subbaiah, C.V. Enhanced Suspicious URL Detection in IoT Using an Optimized Hybrid Selection Technique. *Int. J. Comput. Learn. Intell.* **2025**, *4*, 432–439.
40. Gupta, S.; Barigidad, S.; Hussain, S.; Dubey, S.; Kanaujia, S. Hybrid Machine Learning for Feature-Based Spam Detection. In *Proceedings of the 2025 2nd International Conference on Computational Intelligence, Communication Technology and Networking (CICTN)*; IEEE: New York, NY, USA, 2025; pp. 801–806.
41. Nuiaa Al Ogaili, R.R.; Mahdi, M.I.; Neamah, A.F.; Alradha Alsaidi, S.A.A.; Alsaeedi, A.H.; Dashoor, Z.A.; Manickam, S. PhishNetVAE Cybersecurity Approach: An Integrated Variational Autoencoder and Deep Neural Network Approach for Enhancing Cybersecurity Strategies by Detecting Phishing Attacks. *Int. J. Intell. Eng. Syst.* **2025**, *18*, 59.

42. Opara, C.; Ahmad, M.; Mirjalili, S.; Faris, H.; Aljarah, I.; Kumar, R.; Shukla, P.K. Look before you leap: Detecting phishing web pages by exploiting raw URL and HTML characteristics. *Expert Syst. Appl.* **2024**, *224*, 119883. [CrossRef]
43. Ahmad, M.; Almomani, A.; Aljarah, I.; Faris, H.; Mirjalili, S. PD-UHD features: Phishing Detection Approach using uncooked URL, HTML content, and Domain Name Features. *Sci. Rep.* **2023**, *13*, 2488.
44. Kavya, S.; Sumathi, D. Multimodal and Temporal Graph Fusion Framework for Advanced Phishing Website Detection. *IEEE Access* **2025**, *13*, 74128–74136. [CrossRef]
45. Banerjee, S.; Bhattacharya, A.; Ghosh, R.; Singh, R.; Ghosh, P. Multimodal and Temporal Graph Fusion for Phishing Website Detection. In *Proceedings of the 17th International Conference on COMSNETS*; Accepted COMSNETS Conference Paper; IEEE: New York, NY, USA, 2025.
46. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv* **2019**, arXiv:1910.01108.

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.