Article

Phishing Webpage Detection via Multi-Modal Integration of HTML DOM Graphs and URL Features Based on Graph Convolutional and Transformer Networks

Jun-Ho Yoon [1], Seok-Jun Buu [1,*] and Hae-Jung Kim [2,*]

[1] Department of Computer Engineering, Gyeongsang National University, Jinju-si 52828, Republic of Korea; 2023210465@gnu.ac.kr

[2] Department of Computer Engineering, Kyungil University, Gyeongsan-si 38428, Republic of Korea

[*] Correspondence: sj.buu@gnu.ac.kr (S.-J.B.); hjkim325@kiu.kr (H.-J.K.)

Abstract: Detecting phishing webpages is a critical task in the field of cybersecurity, with significant implications for online safety and data protection. Traditional methods have primarily relied on analyzing URL features, which can be limited in capturing the full context of phishing attacks. In this study, we propose an innovative approach that integrates HTML DOM graph modeling with URL feature analysis using advanced deep learning techniques. The proposed method leverages Graph Convolutional Networks (GCNs) to model the structure of HTML DOM graphs, combined with Convolutional Neural Networks (CNNs) and Transformer Networks to capture the character and word sequence features of URLs, respectively. These multi-modal features are then integrated using a Transformer network, which is adept at selectively capturing the interdependencies and complementary relationships between different feature sets. We evaluated our approach on a real-world dataset comprising URL and HTML DOM graph data collected from 2012 to 2024. This dataset includes over 80 million nodes and edges, providing a robust foundation for testing. Our method demonstrated a significant improvement in performance, achieving a 7.03 percentage point increase in classification accuracy compared to state-of-the-art techniques. Additionally, we conducted ablation tests to further validate the effectiveness of individual features in our model. The results validate the efficacy of integrating HTML DOM structure and URL features using deep learning. Our framework significantly enhances phishing detection capabilities, providing a more accurate and comprehensive solution to identifying malicious webpages.

Keywords: phishing webpage detection; graph convolutional network; transformer network; multi-modal integration; cyberspace security

check for updates

Citation: Yoon, J.-H.; Buu, S.-J.; Kim, H.-J. Phishing Webpage Detection via Multi-Modal Integration of HTML DOM Graphs and URL Features Based on Graph Convolutional and Transformer Networks. *Electronics* **2024**, *13*, 3344. https://doi.org/10.3390/electronics13163344

Academic Editors: Li Pan, Ning Liu and Conghui Zheng

Received: 25 July 2024
Revised: 17 August 2024
Accepted: 20 August 2024
Published: 22 August 2024

1. Introduction

The rapid expansion of internet services has revolutionized how individuals and organizations communicate, conduct transactions, and access information [1–3]. However, this growth has also led to an increase in cybersecurity threats, with phishing attacks becoming one of the most widespread and serious forms of online fraud. Phishing attacks involve creating fake websites that appear to be benign, in order to steal sensitive information such as passwords, credit card numbers, and personal identification details from users [1,4]. As phishing techniques become increasingly sophisticated, detecting these fraudulent activities has become more challenging [5–7].

Phishing websites often attempt to resemble the URLs of benign websites. For example, phishing websites may use domains or paths similar to those of benign websites to deceive users. In such cases, it is difficult to detect phishing solely by analyzing the URL. However, while phishing websites may imitate the URLs of benign websites, they rarely replicate the HTML structure completely [8–10]. By modeling the dependencies within the HTML DOM structure using Graph Convolutional Networks (GCNs) and capturing both character and word-level features of URLs with Convolutional Neural Networks (CNNs) and Transformer Networks, our method aims to provide a more comprehensive and accurate detection mechanism. This approach not only improves detection accuracy but also enhances robustness against sophisticated phishing techniques, making it a more reliable solution for cybersecurity [2,11,12].

In this study, we propose a novel multi-modal approach that combines URL analysis and HTML DOM structure analysis to significantly enhance phishing detection accuracy. Our approach leverages Graph Convolutional Networks (GCNs) to model the complex dependencies within the DOM structure, and Convolutional Neural Networks (CNNs) and Transformer Networks to analyze URL features at both the character and word levels. This multi-modal integration allows our method to capture both the structural and sequential characteristics of phishing websites, making it more robust against sophisticated phishing techniques.

To illustrate the effectiveness of our approach, Figure 1 visualizes how key features, extracted using different models, distinguish between benign and phishing websites. Specifically, Figure 1a–c shows the URL character-level analysis, Figure 1d–f shows URL word-level analysis, and Figure 1g–i depicts HTML tag name analysis. These features are critical inputs to our deep learning model and play a pivotal role in accurately detecting phishing websites. The visualization underscores the importance of these selected features in distinguishing between benign and phishing websites, highlighting the superior detection capability of our model.

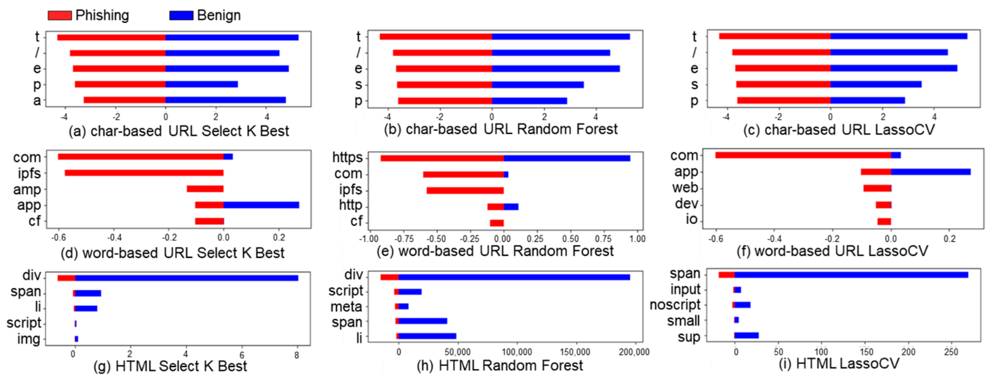

Figure 1. Visualization of key features for phishing and benign classification using three models: Select K Best, Random Forest, and LassoCV (**a**–**c**) show URL character-level analysis, (**d**–**f**) show URL word-level analysis, and (**g**–**i**) show HTML tag name analysis.

Moreover, our approach demonstrates that phishing websites, while often mimicking the URL patterns of benign websites, do not replicate the HTML structure as effectively. For instance, as shown in Table 1 and Figure 2, both URLs follow a similar pattern of "script.google.com", but the HTML DOM structures differ significantly. Case (a) represents a benign site with a well-organized HTML structure, whereas Case (b), a phishing site, has a more simplified and irregular structure. This difference in HTML complexity provides crucial information for our model to identify phishing sites, even when the URL appears legitimate.

Table 1. Ground truth labels and corresponding URLs for the HTML DOM structures illustrated in Figure 2. Case (a) represents a benign site with a standard URL, while Case (b) shows a phishing site that mimics the benign URL but has a different, more simplified HTML structure.

Case	Ground Truth	URL
(a)	Benign	https://script.google.com (accessed on 19 August 2024)
(b)	Phishing	https://script.google.com/macros/s/AKfycbyjt18r7uGwlzNe6SekQE0yCNgfYHE5mHe1ib-9SIKfuT0KKTu8oaCrdXoXhbg3ixjl/exec (accessed on 19 August 2024)

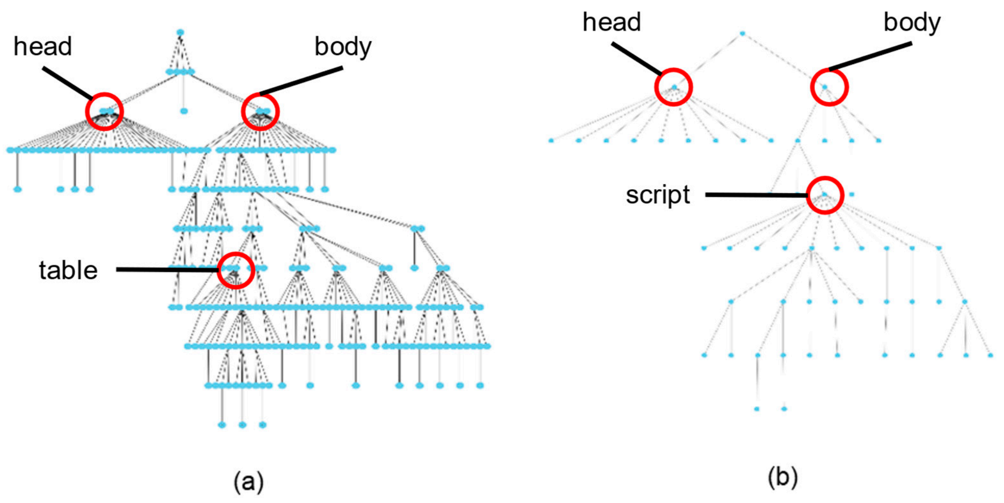

Figure 2. Comparison of HTML DOM structures between (**a**) a benign website and (**b**) a phishing website with similar URLs.

Our method has been validated using a large-scale real-world dataset and has achieved a 7.03% improvement in classification accuracy compared to existing state-of-the-art techniques. These results demonstrate that a deep learning-based approach, which combines HTML DOM structure and URL features, plays a crucial role in enhancing phishing detection capabilities.

2. Related Works

Phishing detection has long been a critical area of cybersecurity research. As summarized in Table 2, various approaches have been developed over the years to tackle this issue. Early approaches mainly focused on analyzing URL features, considering factors like URL length, suspicious substrings, and domain reputation. For instance, the Texception model uses convolutional layers to analyze both character-level and word-level information of URLs, achieving notable performance on large datasets [8]. Advancements in phishing detection have seen the integration of multiple machine learning techniques, such as MOE/RF, which combines multi-objective evolution optimization with Random Forest, yielding high accuracy and recall [13]. Similarly, GramBeddings employs a four-channel architecture with CNN, LSTM, and attention layers, demonstrating significant accuracy on various datasets [14]. The use of adversarial examples, as seen in URLBUG, highlights the challenges posed by adversarial attacks, which degrade the performance of machine learning models. Notably, URLBUG's performance was lower compared to other models because it tested on adversarial URLs generated to deliberately evade detection, showcasing the difficulties in maintaining robustness when dealing with generated data [15]. For example, a method combines multiple machine learning techniques to analyze the lexical features of URLs and web-scraped content, integrating URL structure and web content for a more comprehensive detection approach by capturing diverse phishing indicators [16]. Another method explores embedding URL components and testing against adversarial attacks, enhancing model robustness and making it more effective in responding to sophisticated evasion techniques [10].

While URL-based methods offer valuable insights, they often fail to capture the full context of phishing attacks. HTML-based approaches, such as PhishSim, address this limitation by analyzing the content and structure of the webpage, achieving high detection rates [17]. Recent research has increasingly focused on integrating URL and HTML features for a more comprehensive detection strategy. For example, a method integrates MLP for structured data with NLP models for HTML content, fusing embeddings to improve detection accuracy [18].

Table 2. Overview of recent studies on phishing detection, categorized by URL-based, HTML-based, and multimodal approaches, including methodologies, data representations, datasets, and performance metrics.

Approach	Representation	Description	Dataset	Performance
URL Classification	Character-level, Word-level [8]	Uses parallel convolutional layers to analyze character and word-level URL information	Collected 1.7 M samples from Microsoft browsing telemetry data	TPR: 47.95% Error Rate: 0.28%
	URL features [13]	Combines multi-objective evolution optimization with Random Forest for phishing detection	Five different URL datasets (Kaggle, Mendeley Data)	Accuracy: 99.04% Recall: 99.48%
	N-gram embeddings [14]	Utilizes n-gram embeddings with a four-channel architecture, includes CNN, LSTM, attention layers	Alexa, Majestic (benign): 400 K Phishtank, Openphish (phishing): 400 K	Accuracy: 98.27% F1 Score: 98.26%
	Adversarial URL Generation [15]	A method to generate adversarial URL by obfuscating domain, path, and TLD parts of URL to test the robustness of ML-based phishing URL detectors	Dataset consists of five different sources with a total of 193,386 benign: 96,693 phishing: 96,693	(Performance reduction occurred due to testing on generated adversarial data) Accuracy: −41.62% F1 Score: −59.17%
	Lexical, Web-scrapped [16]	Combines multiple ML techniques to extract and analyze lexical and web-scrapped features	Dataset consists of 12 different sources with a total of 3,980,870	Accuracy: 99.63% Precision: 99.60%
	URL embedding [10]	Analyzes robustness of ML models against adversarial attacks	Alexa (benign): 10,000 Phishtank (phishing): Not specified	Precision: 91% Recall: About 93% F1 Score: About 92.5%
HTML Classification	HTML content [17]	Uses Normalized Compression Distance to compare HTML content with known phishing pages	Common Crawl (benign): 180,302 Phishtank (phishing): 9034	AUC: 98.68% TPR: About 90% FPR: 0.58%
	HTML content [18]	Integrates MLP for structured data with NLP models for HTML content, fuses embeddings	Alexa (benign): 2000 Openphish (phishing): 2000	Accuracy: 97.18% F1 Score: 96.80%
Multimodal Classification	Raw URL, HTML content [19]	Combines raw URL and HTML content analysis using embeddings and convolutional layers	Alexa (benign): 22,687 Phishtank (phishing): 22,687	Accuracy: 98.1% Precision: 98.2% Recall: 98.1% F1 Score: 98.1%
	Image, Raw URL, HTML Tags [9]	Combines raw URL and HTML tags and image analysis using word embeddings and convolutional layers	Collected from PhishiTank and OpenPhish benign: 8316 phishing: 7848	Accuracy: 95.35% Precision: 94.39% Recall: 94.26% F1 Score: 94.34%
	URL, HTML [20]	Proposes PhiUSIIL, a phishing URL detection framework combining URL similarity indexing and incremental learning for real-time threat adaptation	Open PageRank Initiative Anon (benign): 134,850 Phishtank, OpenPhish, MalwareWorld (phishing): 100,945	Accuracy: 99.97% Precision: 99.97% Recall: 99.98% F1 Score: 99.98%

Additionally, the WebPhish framework combines raw URL and HTML content analysis, achieving high accuracy and demonstrating the effectiveness of multimodal approaches [19]. The PhiUSIIL framework leverages URL similarity indexing and incremental learning to adapt to real-time threats, achieving near-perfect accuracy, precision, and recall, which underscores the potential of real-time adaptive models in phishing detection [20]. Another notable approach integrates raw URL, HTML tags, and image analysis using word embeddings and convolutional layers, achieving high accuracy and demonstrating the benefits of incorporating multiple data types for phishing detection [9].

Our approach distinguishes itself from prior research by leveraging Graph Convolutional Networks to effectively model the complex dependencies among HTML tags within the DOM structure, thereby optimizing feature representation for phishing detection. Furthermore, we employ a Transformer network to integrate URL features with HTML DOM Graph features, enabling the model to selectively attend to and extract complementary relationships among these multi-modal features. This precise feature integration enhances the overall detection accuracy and robustness.

3. Proposed Method

In this section, we present a phishing detection framework that integrates multi-modal features from HTML DOM Graphs and URL features using Graph Convolutional and Transformer Networks. As shown in Figure 3, our approach combines different deep learning models to achieve a more accurate classification of phishing webpages.

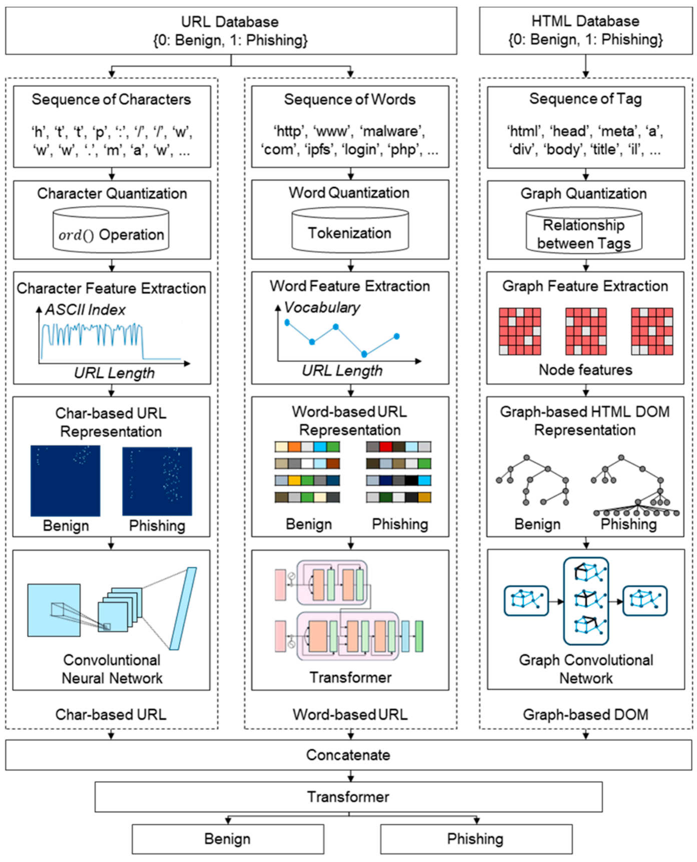

Figure 3. Overview of the proposed phishing detection framework integrating HTML DOM graphs and URL features.

3.1. URL and HTML Data Representation for Phishing Detection

To effectively detect phishing webpages, it is essential to analyze and represent the data from multiple perspectives. This section outlines our approach for representing both URL and HTML data, which is crucial for identifying patterns and characteristics unique to phishing attempts. Our method uses deep learning techniques to process and extract features from URLs and HTML content, providing a robust framework for phishing detection.

3.1.1. Char-Based URL Feature Extraction

The character-based URL feature extraction involved converting each URL into a matrix representation using one-hot encoding. Each character in a URL was mapped to its corresponding ASCII value, resulting in a numerical representation. As outlined in Algorithm 1, the ASCII values were then converted into a 128 × 128 one-hot encoded matrix, where each row represented a character, and each column corresponded to a possible ASCII value. This process ensured that the URL was represented as a fixed-size input suitable for CNN processing. The CNN architecture employed consisted of three convolutional layers with ReLU activation functions and max-pooling layers to capture local dependencies and spatial hierarchies in character sequences. To summarize, the URL string is first converted into a list of ASCII values. These values are then padded or truncated to a fixed length of 128 characters to maintain uniformity across all URLs. The padded ASCII values are finally one-hot encoded into a 128 × 128 matrix.

Algorithm 1: Char-based URL Representation for Phishing Detection

```
1: Input: A URL string
2: Output: A 128×128 matrix representing the URL
3: function ConvertToASCII(url)
4:       return [ord(char) for char in url]
5: end function
6: function ApplyPadding(ascii_values)
7:      max_length ← 128
8:      return (ascii_values[:max_length] + [0] × (max_length − len(ascii_values)))[:max_length]
9: end function
10: function OneHotEncode(ascii_values)
11:      matrix ← zero matrix of size 128 × 128
12:      for i, value in enumerate(ascii_values) do
13:           matrix[i][value] ← 1
14:      end for
15:      return matrix
16: end function
17: ascii_values ← ConvertToASCII(URL)
18: padded_values ← ApplyPadding(ascii_values)
19: matrix ← OneHotEncode(padded_values)
```

Given this representation, we proceed with feature extraction using CNNs. Convolutional Neural Networks (CNNs) are a class of deep neural networks specifically designed to process data with a grid-like topology, such as images or sequences of characters. They are particularly effective for tasks involving spatial hierarchies and pattern recognition. In the context of char-based URLs feature extraction, CNNs are used to capture local patterns and dependencies among characters in the one-hot encoded URL matrix. By applying convolutional filters across the matrix, CNNs can identify and learn important character combinations and sequences that may indicate phishing attempts. The convolutional operation is given by Equation (1).

$$\mathbf{H}_l = f(\phi(\mathbf{W}_l, \mathbf{X}) + \mathbf{b}_l) \tag{1}$$

where $\mathbf{H}_l$ is the output feature map of layer l, $\mathbf{W}_l$ represents the convolutional filter weights, ϕ denotes the convolution operation, $\mathbf{b}_l$ is the bias term, and $f(\cdot)$ is the activation function.

The key advantage of CNNs in this application is their ability to automatically learn and detect patterns that are spatially invariant, making them well-suited for identifying phishing URLs where certain character sequences or patterns might recur across different URLs.

To train the CNN model, we define the loss function as the categorical cross-entropy loss, which measures the discrepancy between the predicted probabilities and the true labels. The loss function $\mathcal{L}_{CNN}$ is given by Equation (2).

$$\mathcal{L}_{CNN} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} log(\sigma(\mathbf{W}_O\mathbf{H}_L(\mathbf{X}_i) + \mathbf{b}_O)_c) \tag{2}$$

where N is the number of URL samples. C is the number of classes. $y_{i,c}$ is the true label of the i-th sample for class c, $\mathbf{X}_i$ is the i-th input sample. $\mathbf{H}_L(\mathbf{X}_i)$ is the feature map obtained from the last layer $\mathcal{L}_{CNN}$ of the CNN for the i-th input char-based URL. $\mathbf{W}_O$ and $\mathbf{b}_O$ are the weights and biases of the output layer, respectively. σ is the softmax function. $p_{i,c}$ is the predicted probability of the i-th input belonging to class c, computed as $\sigma(\mathbf{W}_O\mathbf{H}_L(\mathbf{X}_i) + \mathbf{b}_O)_c$.

3.1.2. Word-Based URL Feature Extraction

Word-based URL feature extraction commenced with the segmentation of URLs into tokens using special characters such as slashes (/), dots (.), and hyphens (-) as delimiters. These tokens were further refined by filtering out common stopwords and retaining only those with semantic significance. Subsequently, a frequency analysis was performed on the tokens to construct a vocabulary dictionary containing the top 5000 most frequently occurring words, each mapped to a unique integer identifier. The tokenized URLs were transformed into sequences of integers and padded to a uniform length of 20 tokens to facilitate batch processing. These integer sequences served as inputs to the Transformer network, which consisted of an embedding layer, a multi-head attention mechanism with two attention heads, and a position-wise feed-forward network. The attention mechanism computed contextual relevance by evaluating the importance of each token within the sequence, allowing the model to capture long-range dependencies effectively.

To represent URLs as sequences of words effectively, we first split each URL string using a set of predefined special characters. This splitting process converts the URL into a list of words. Next, we build a dictionary of the most frequently occurring words across the dataset, mapping each word to a unique integer. The URLs are then transformed into sequences of these integers, which can be processed by Transformer Networks to capture contextual dependencies among the words.

Algorithm 2 describes the detailed steps for converting URLs into word-based sequences of integers. Following this preprocessing, the transformed URL sequences are fed into a Transformer Network to capture the contextual relationships and dependencies among the words. Transformer Networks are particularly well-suited for this task as they excel in modeling long-range dependencies and capturing intricate patterns within sequential data. In the context of word-based URLs feature extraction, we utilize the Transformer's ability to understand the contextual meaning of words within a URL by employing self-attention mechanisms. The attention mechanism, as shown in Equation (3), computes the relevance of different parts of the sequence, allowing the model to weigh the importance of each token dynamically.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{3}$$

where Q, K, V are the query, key, and value matrices, and d_k is the dimension of the key vectors.

Algorithm 2: Word-based URL Representation for Phishing Detection

```
1: Input: A list of URL
2: Output: Transformed URL into sequences of integers
3: Initialize an empty dictionary D
4: Define special characters S ← "!@#%∧&*() +-=[]{}\—'"",./<>?"
5: function SplitURL(url)
6:         return split(url, S)
7: end function
8: function BuildDictionary(words)
9:         Count and sort words by frequency
10:         return top 5000 words with indices 1 to 5,000
11: end function
12: function TransformURL(url, D)
13:         words ← SplitURL(url)
14:         sequence ← [D[word] if word ∈ D else 0 for word in words]
15:         return (sequence[:20] + [0] × 20)[:20]
16: end function
17: all_words ← flatten(SplitURL(url) for url in list_of_URL)
18: D ← BuildDictionary(all_words)
19: all_sequences ← [TransformURL(url, D) for url in list_of_URL]
```

For a given input sequence $\mathbf{X}$, the multi-head attention is defined as Equation (4).

$$\begin{gathered}\text{MultiHead}(Q,K,V) = \text{Concat}(head_1,\ldots,head_h)W^O \\ head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)\end{gathered} \tag{4}$$

where W_i^Q, W_i^K, W_i^V are learned projection matrices. The output of the multi-head attention is then passed through a feed-forward neural network. Let x be the output of the multi-head attention mechanism.

The feed-forward neural network is defined as Equation (5).

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{5}$$

where W_1, W_2, b_1, b_2 are learned parameters. The final output from the Transformer model is given by $\mathbf{Z} = \text{Transformer}(\mathbf{X})$, which represents the high-level features extracted from the input sequence. The Transformer function is defined as Equation (6).

$$\text{Transformer}(\mathbf{X}) = FFN(\text{MultiHead}(Q,K,V)) \tag{6}$$

These high-level features extracted by the Transformer are then fed into a final classification layer to detect phishing webpages. To train the Transformer model, we define the loss function as the categorical cross-entropy loss, which measures the discrepancy between the predicted probabilities and the true labels. The loss function $\mathcal{L}_{\text{Transformer}}$ is given by Equation (7).

$$\mathcal{L}_{\text{Transformer}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} log(\sigma(\mathbf{W}_O\mathbf{Z}_i + \mathbf{b}_O)_c) \tag{7}$$

where $\mathbf{X}_i$ represents the i-th input sequence, $\mathbf{Z}_i$ is the output of the Transformer for the i-th input sequence. $\mathbf{W}_O$ and $\mathbf{b}_O$ are the weights and biases of the output layer, respectively, σ is the softmax function, and $p_{i,c}$ is the predicted probability of the i-th sample belonging to class c, computed as $\sigma(\mathbf{W}_O\mathbf{Z}_i + \mathbf{b}_O)_c$.

3.1.3. DOM Graph Feature Extraction

The HTML documents were parsed to extract the DOM tree structure using BeautifulSoup 4.12.3, a Python 3.9 library for web scraping. Each HTML element was represented as a node, and edges were established based on parent–child relationships in the DOM hierarchy. The graph representation captured both the element types and their relational context, crucial for understanding the structural nuances of phishing pages. The HTML content of each webpage was first parsed into a DOM tree using BeautifulSoup. Each HTML element (e.g., <div>, <a>, <p>) was treated as a node in this tree. The hierarchical relationships between these elements were preserved, with parent–child relationships represented as edges in the graph. Each node in the graph represented an HTML tag, and attributes of these tags (such as id, class, and href attributes) were initially considered as potential features. However, to maintain computational efficiency and focus on structural aspects, the node features were limited to the tag names and a simplified attribute representation, such as the presence or absence of key attributes (e.g., href, src).

The process begins with parsing the HTML document to construct the DOM tree. This tree structure is then converted into a graph, where nodes represent HTML elements, and edges represent parent–child relationships between these elements. The detailed steps for this transformation are provided in Algorithm 3. Once the DOM tree is represented as a graph, we proceed to feature extraction using Graph Convolutional Networks (GCNs). GCNs are well-suited for this task as they excel in capturing the relational and structural information inherent in graph data. The following equations outline the operations involved in applying GCNs to the graph representation of the HTML DOM graph.

Algorithm 3: Graph-based HTML DOM Graph Representation for Phishing Detection

```
1: Input: An HTML document
2: Output: A graph representation of the HTML DOM Tree
3: function ParseHTML(html document)
4:      dom tree ← parse html document into DOM Tree
5:      return dom tree
6: end function
7: function BuildGraph(dom tree)
8:      if dom tree is empty then
9:          return 0                    # Return 0 if no nodes or edges exist
10:     end if
11:     Initialize graph as an empty graph
12:     Use a queue to perform level-order traversal of dom tree
13:     while queue is not empty do
14:         node ← dequeue()
15:         Add node to graph as a vertex
16:         for each child of node do
17:             Add child to graph as a vertex
18:             Add an edge from node to child in graph
19:         end for
20:     end while
21:     return graph
22: end function
23: dom tree ← ParseHTML(HTML document)
24: graph ← BuildGraph(dom tree)
```

In the context of DOM tree feature extraction, Graph Convolutional Networks (GCNs) extend the concept of convolutional networks to graph-structured data. GCNs effectively learn node representations by aggregating features from neighboring nodes. This capability allows GCNs to capture complex relationships within the HTML DOM tree by leveraging both the structure of the graph and the attributes of its nodes. The graph convolutional operation is given by Equation (8).

$$\mathbf{H}(l+1) = ReLU\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right) \tag{8}$$

where $\hat{A} = A + I$ is the adjacency matrix with added self-connections, $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$, $\mathbf{H}^{(l)}$ is the matrix of activations in the l-th layer, $\mathbf{W}^{(l)}$ is the weight matrix of the l-th GCN layer, and ReLU is the activation function applied element-wise.

The initial input to the GCN, $\mathbf{H}^{(0)}$, is the feature matrix $\mathbf{X}$ derived from node attributes. The output of the final GCN layer $\mathbf{H}^{(L)}$ serves as the high-level feature representation of the graph.

To train the GCN model, we use the categorical cross-entropy loss function, which measures the discrepancy between the predicted class probabilities and the true labels. The loss function $\mathcal{L}_{GCN}$ is defined as Equation (9).

$$\mathcal{L}_{\text{GCN}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} log\left(\sigma\left(\mathbf{W}_O\mathbf{H}^{(L)}(\mathbf{X}_i) + \mathbf{b}_O\right)_c\right) \tag{9}$$

where N is the number of URL samples, C is the number of classes, $y_{i,c}$ is the true label of the i-th sample for class c, σ is the softmax function, $\mathbf{X}_i$ is the i-th input graph sample, and $\mathbf{H}^{(L)}(\mathbf{X}_i)$ is the output from the last GCN layer for the i-th input graph. The use of GCNs significantly enhances phishing detection accuracy by effectively leveraging the structural information inherent in HTML DOM graphs.

3.2. Ensemble Classifier Utilizing Char-Based URL, Word-Based URL, and HTML DOM Graph Features

In this section, we propose a multimodal ensemble model for phishing detection. Various features can be utilized to detect phishing attacks, such as URL, SSL certificate, HTML DOM, webpage content, HTML header, and protocol, among others [1,4,7,21]. However, many of these features can be manipulated to appear benign, potentially deceiving both users and phishing detection systems [4]. The key point, as argued by Wang et al. [22], is that the most influential feature in determining whether a case is phishing varies for each instance [23–29]. Therefore, it is crucial to leverage as many features as possible simultaneously while excluding those susceptible to manipulation that might hinder learning [22,28–34]. This approach maximizes the phishing detection capability.

We introduce an ensemble classification model that integrates the strengths of CNNs for char-based URL features, Transformers for word-based URL features, and GCNs for graph-based HTML DOM features. This multi-modal approach aims to leverage the complementary nature of these different feature representations to enhance the overall accuracy and robustness of phishing detection. The architecture of the ensemble model is illustrated in Figure 4.

The architecture of our proposed model was designed to effectively integrate multiple data modalities for phishing detection, combining URL and HTML DOM features. The selection of model components and hyperparameters was guided by empirical experimentation and best practices in the field of deep learning. We employed Convolutional Neural Networks (CNNs) to process character-based URL features, leveraging their strength in capturing local spatial hierarchies. The architecture includes two convolutional layers with kernel sizes of 3×3, chosen for their ability to detect small patterns and variations in character sequences. The ReLU activation function was applied to introduce non-linearity, and max-pooling layers were used to reduce dimensionality and computational cost. We utilized Transformer networks for word-based URL analysis due to their superior ability to model long-range dependencies and context within sequences. The model included two attention heads, providing a balance between computational efficiency and the ability to capture complex relationships within word sequences. Positional encoding was crucial for maintaining sequence order, enhancing the model's understanding of contextual word relationships. We applied Graph Convolutional Networks (GCNs) to capture the relational information inherent in the HTML DOM structure. The architecture consisted of two GCN layers to aggregate information from neighboring nodes effectively, with the number of features per node optimized to ensure a good trade-off between model complexity and accuracy. The ReLU activation function, consistent with that used in CNNs, was utilized to ensure efficient learning across layers.

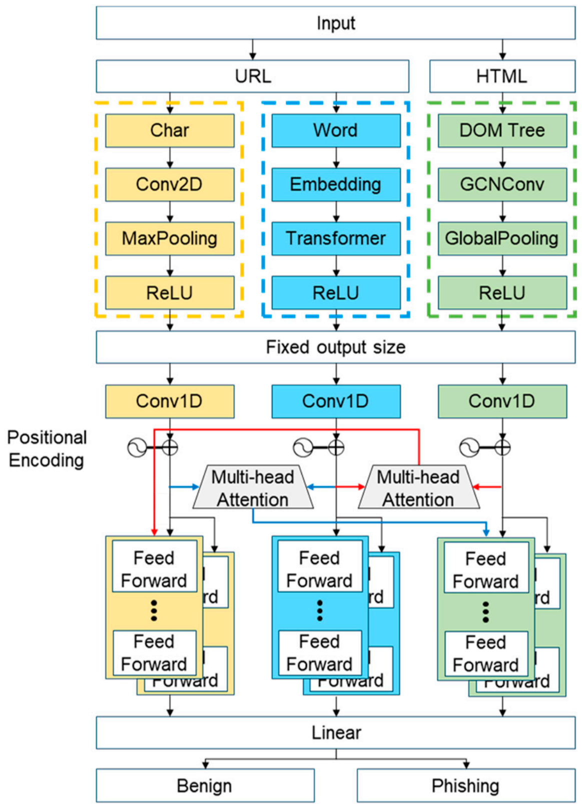

Figure 4. The ensemble classification model combining CNN, Transformer, and GCN components for phishing detection.

The ensemble classifier consists of three primary components. First, the CNN component processes the one-hot encoded URL character matrix and extracts local patterns and dependencies among characters. The output is defined as $\mathbf{H}_{\text{char}} = \text{CNN}(\mathbf{X}_{\text{char}})$. Second, the Transformer component transforms URLs into sequences of integers based on word tokens and captures long-range dependencies and contextual relationships among words. The output is defined as $\mathbf{H}_{\text{word}} = \text{Transformer}(\mathbf{X}_{\text{word}})$. Third, the GCN component models the structural and relational information of the HTML DOM tree using graph convolutional operations. The output is defined as $\mathbf{H}_{\text{graph}} = \text{GCN}\left(\mathbf{A}, \mathbf{X}_{\text{graph}}\right)$.

The extracted feature representations from the three components are concatenated to form a comprehensive feature vector:

$$\begin{gathered}\mathbf{H}_{\text{concat}} = \text{Concat}\left(\mathbf{H}_{\text{char}}, \mathbf{H}_{word}, \mathbf{H}_{graph}\right)\\ \mathbf{Z} = \mathbf{W}_{fc} Transformer(\mathbf{H}_{concat}) + \mathbf{b}_{\text{fc}}\\ \boldsymbol{P} = \text{softmax}(\mathbf{Z})\end{gathered} \tag{10}$$

where $\mathbf{W}_{fc}$ and $\mathbf{b}_{fc}$ are the weights and biases of the fully connected layer, $\mathbf{H}_{\text{concat}}$ is the concatenated feature vector, and $\mathbf{P}$ represents the predicted probabilities for each class.

The ensemble model is trained based on the categorical cross-entropy loss function, defined as Equation (11).

$$\mathcal{L}_{CNN} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} log(P_{i,c}) \tag{11}$$

where N is the number of URL samples, C is the number of classes, $y_{i,c}$ is the true label of the i-th sample for class c, and $P_{i,c}$ is the predicted probability of the i-th sample belonging to class c.

By integrating the char-based URL, word-based URL, and HTML DOM graph features and passing them through a Transformer layer, our ensemble model leverages the unique strengths of each feature representation, resulting in a more accurate and robust phishing detection system.

4. Experimental Results

4.1. Dataset and Preprocessing

In this section, we detail the datasets used and the preprocessing steps undertaken to prepare the data for our experiments. Table 3 summarizes the datasets used for validating the proposed method. Our analysis is based on two primary datasets: benign data sourced from Common Crawl and phishing data obtained from Phishtank. These datasets form the foundation of our study.

Table 3. Summary of Datasets Used for Phishing Detection Analysis.

Source	Class	Quantity	Examples
Common Crawl	Benign	60,000	https://www.policyspark...
Phishtank	Phishing	14,912	http://yujjf.duckdns...
Mendeley Data	Phishing	14,573	http://masf.krhes.boston...

To address the significant class imbalance in the benign dataset, which could negatively impact the evaluation of our experimental results, we performed a careful down-sampling. Specifically, the benign dataset was reduced to 60,000 instances to achieve a more balanced dataset, ensuring a fair comparison with the phishing data. This step was essential to maintain the validity of our evaluation and to avoid skewed results that could misrepresent model performance.

We collected a total of 38,060 phishing instances, carefully selected to encompass a wide range of phishing techniques. From this collection, we used BeautifulSoup to parse the HTML content. Through this process, we successfully retrieved the HTML content for 14,912 instances. This subset of successfully parsed phishing data provided a robust foundation for further analysis and modeling, ensuring that our experiments were grounded in high-quality data.

To contribute to the research community and further support advancements in phishing detection, we plan to release this dataset as an open-source benchmark. This dataset, which includes a diverse and comprehensive collection of phishing and benign data, will serve as a valuable resource for effectively benchmarking future phishing detection models. Our goal is to promote innovation and drive progress in this critical field of cybersecurity.

4.2. Implementation Details and Hyperparameter Settings

For the implementation of our experiments, we used the Python deep learning library PyTorch (version 2.0.1) in conjunction with the graph deep learning library Spektral (version 1.3.0), TensorFlow-gpu (version 2.9.0), and Scikit-learn (version 1.3.0) for preprocessing and evaluation purposes. We conducted our experiments on NVIDIA A6000 GPUs.

The CNN component utilizes two 2D convolutional layers with dropout in between to prevent overfitting. The Transformer component includes embedding, positional encoding, dropout, and a transformer encoder layer to capture complex patterns within the data. The GCN component employs GCNConv layers and global mean pooling to effectively model graph-structured HTML DOM data.

The ensemble model integrates these components, using an embedding layer followed by a transformer encoder and a linear layer to combine the strengths of each individual model. This setup allows for capturing diverse features and dependencies, improving the overall phishing detection performance. The specific hyperparameter settings, such as the number of units, activation functions, and parameter counts, are optimized for effective model training and evaluation. Table 4 summarizes the layers and configurations used in the ensemble model, providing a clear overview of the implementation details and parameter settings.

Table 4. Summary of the Proposed Multi-modal Ensemble Model Layers.

Convolutional Neural Network		**Transformer**		**Graph Convolutional Network**	
Operation	No. of Parameters	Operation	No. of Parameters	Operation	No. of Parameters
Convolution 2D	160	Embedding	320,000	GCNConv	6464
Dropout	-	Positional Encoding	-	Dropout	-
Convolution 2D	4640	Dropout	-	GCNConv	2080
Dropout	-	Transformer Encoder	33,472	Global mean pooling	-

Ensemble for Selectively Weighting and Combining Features Based on Transformer			
Operation	No. of Units/Heads	Activation Function	No. of Parameters
Embedding	8	relu	40,000
Positional Encoding	-	-	-
Dropout	-	-	-
Transformer Encoder	2 heads	relu	672
Linear	2	-	66

4.3. Performance Comparison

We evaluate the performance of our proposed ensemble model in comparison to various baseline models and state-of-the-art techniques using 10-fold cross-validation. The evaluation metrics considered are accuracy, precision, recall, and F1 score.

The performance of the base networks reveals several key insights. The Convolutional Neural Network (CNN) demonstrated strong performance with high accuracy and recall, indicating its effectiveness in capturing character-level features from URLs. Similarly, the Transformer model performed well, showcasing its ability to handle sequential dependencies in URLs. However, the Graph Convolutional Network (GCN) showed slightly lower performance compared to the CNN and Transformer, which may be due to the inherent complexity of modeling HTML DOM trees.

Among the comparative studies, the URLNet model outperformed several others, with a notable accuracy and precision, demonstrating the effectiveness of combining multiple URL features. The Texception model, while having a high recall, showed significant variability in its precision, indicating potential challenges in handling diverse phishing tactics. PhishDet, on the other hand, achieved the highest scores across most metrics, affirming its robustness and reliability in phishing detection tasks.

Our proposed ensemble model demonstrated exceptional performance, achieving up to a 22% improvement in precision and up to a 23% improvement in recall compared to baseline models. These results highlight the superiority of our approach, combining the strengths of CNNs, Transformers, and GCNs to create a more comprehensive and effective phishing detection system. The ensemble model's ability to leverage multiple modalities of data significantly enhances its detection accuracy and robustness against various phishing techniques. Table 5 provides a detailed comparison of the performance metrics for all evaluated models, illustrating the effectiveness of our ensemble approach in improving phishing detection accuracy and reliability. The best performance for each metric is highlighted in bold.

Table 5. Performance Comparison of Different Models (10-fold cross-validation).

Method	Accuracy	Precision	Recall	F1 Score
		Base networks		
CNN	0.8952 ± 0.0190	0.8626 ± 0.0205	0.8952 ± 0.0190	0.8736 ± 0.0204
Transformer	0.8868 ± 0.0366	0.8859 ± 0.0678	0.8868 ± 0.0366	0.8856 ± 0.0560
GCN	0.8739 ± 0.0091	0.8684 ± 0.0086	0.8740 ± 0.0090	0.8605 ± 0.0144
URLDet [35]	0.8014 ± 0.0214	0.4007 ± 0.1042	0.5089 ± 0.1189	0.4959 ± 0.0510
HTMLDet [35]	0.9154 ± 0.0292	0.8784 ± 0.0221	0.8416 ± 0.0301	0.8596 ± 0.0255
		Comparative studies		
URLNet [36]	0.9425 ± 0.0208	0.9033 ± 0.0785	0.8051 ± 0.1195	0.8453 ± 0.0637
Texception [8]	0.8534 ± 0.1308	0.8457 ± 0.1536	0.9714 ± 0.0286	0.8889 ± 0.0954
WebPhish [19]	0.9290 ± 0.0719	0.9336 ± 0.0353	0.8689 ± 0.1311	0.8937 ± 0.1386
PhishDet [35]	**0.9853 ± 0.0058**	0.9522 ± 0.0204	0.9721 ± 0.0070	0.9620 ± 0.0091
		Ours		
Ours	0.9812 ± 0.0033	**0.9658 ± 0.0125**	**0.9765 ± 0.0042**	**0.9709 ± 0.0050**

4.4. Hyperparmeter Impact Analysis

In this section, we explore the influence of different hyperparameters on the performance of our proposed phishing detection model. One of the critical hyperparameters in our model is the number of heads used in the Multi-Head Attention mechanism. Multi-Head Attention allows the model to focus on different parts of the input data simultaneously, providing a richer and more diverse representation. We conducted experiments using four different numbers of attention heads: 4, 8, 16, and 32, to determine the optimal configuration for our model. Table 6 presents the results of these experiments, showing the accuracy, precision, recall, and F1 score for each configuration. The best performance for each metric is highlighted in bold. The results indicate that using eight attention heads achieved the highest overall performance across all metrics, with an accuracy of 0.9884, precision of 0.9916, recall of 0.9938, and an F1 score of 0.9927. This suggests that using eight heads offers a good balance between model complexity and the ability to capture diverse aspects of the input data, leading to more accurate and reliable phishing detection.

Table 6. Performance metrics for different batch sizes during model training.

No. of Heads	Accuracy	Precision	Recall	F1 Score
4	0.9851	0.9600	**1.0000**	0.9796
8	**0.9884**	0.9916	0.9938	**0.9927**
16	0.9371	**1.0000**	0.9239	0.9610
32	0.9732	1.0000	0.8250	0.9041

4.5. Ablation Study

In this section, we present the results of our ablation study to evaluate the importance of incorporating character-based URL, word-based URL, and HTML DOM graph features in our phishing detection model. Table 7 summarizes the performance metrics (accuracy, precision, recall, and F1 score) for various configurations of our model, where different combinations of the three feature types are used. The observation of the highest performance when using all three features together suggests that incorporating each feature is crucial for effective phishing detection.

Table 7. Performance Metrics for Different Feature Combinations in the Ablation Study.

Data			Accuracy	Precision	Recall	F1 Score
Char-Based URL	**Word-Based URL**	**HTML DOM Graph**				
✔			0.8952	0.8626	0.8952	0.8736
	✔		0.8868	0.8859	0.8868	0.8856
		✔	0.8739	0.8684	0.8740	0.8605
✔	✔		0.9540	0.9167	0.8544	0.8844
✔		✔	0.9817	0.9436	0.9647	0.9541
	✔	✔	0.9789	0.9245	0.9730	0.9481
✔	✔	✔	**0.9884**	**0.9677**	**0.9759**	**0.9718**

4.6. t-SNE Visualization of Feature Integration

In this section, we present t-Distributed Stochastic Neighbor Embedding (t-SNE) visualizations to illustrate the effectiveness of integrating different features for phishing detection. The t-SNE algorithm reduces the dimensionality of our feature space, enabling a clearer visual comparison of the feature distributions. Figure 5 depicts the t-SNE plots for different feature combinations. Figure 5a represents the data distribution using the URLNet method, which relies solely on character-based and word-based URL features. The data points are scattered with some clustering, indicating that while URLNet captures certain phishing characteristics, it lacks robustness due to its exclusive reliance on URL-based features. In Figure 5b, the distribution is illustrated using the Texception method, which also depends on word-based URL features. The method shows some clustering but falls short of achieving optimal phishing detection accuracy, likely due to its limited feature set. Figure 5c presents the data distribution using the proposed method, which utilizes only character-based and word-based URL features, deliberately excluding the HTML DOM graph structure. However, the absence of HTML structure limits the model's ability to fully separate benign and phishing instances.

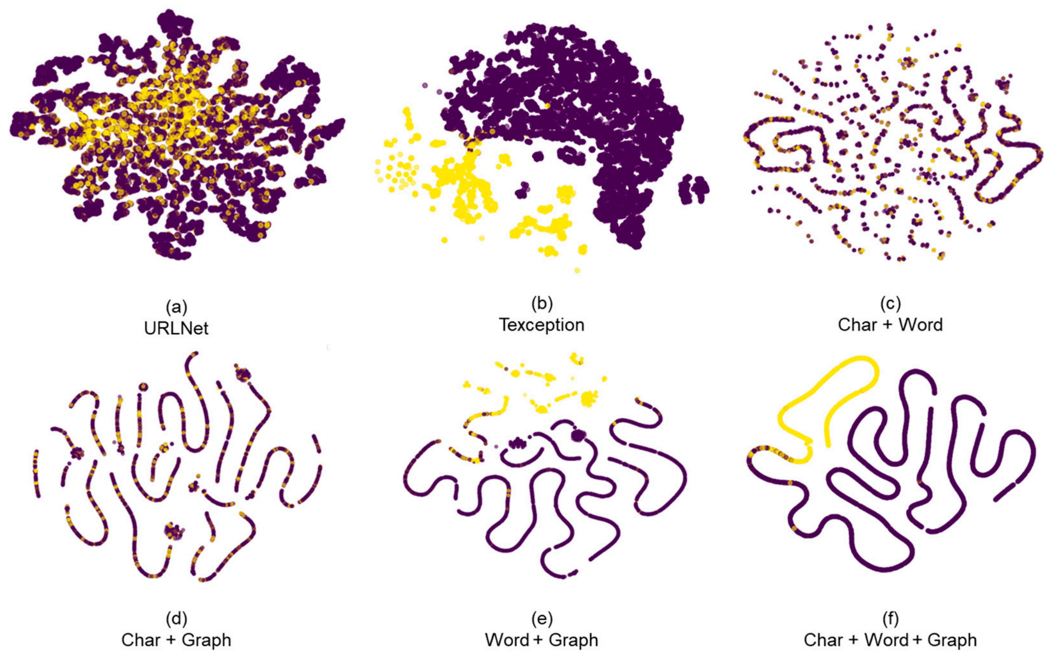

Figure 5. t-SNE Visualization of Feature Integration for Graph-Based and Non-Graph-Based Models.

Figure 5d combines character-based URL features with the HTML DOM graph structure as part of the proposed method. The clustering is distinct, demonstrating the effectiveness of integrating these URL features with the HTML structure. However, the clusters are not completely separated, indicating that while the integration improves detection, it is not yet optimal. Similarly, Figure 5e combines word-based URL features with the HTML DOM graph structure using the proposed method. The clustering is more defined compared to previous cases, showing improved separation between benign and phishing instances. This result underscores the value of combining word-based URL features with the HTML DOM graph. Finally, Figure 5f employs the proposed method with all three features: character-based URL, word-based URL, and HTML DOM graph. The clear and well-defined separation between clusters highlights the importance of integrating all three feature types for achieving the most accurate phishing detection.

The comparison between these plots underscores the critical importance of using all three feature types together. The clear separation seen in Figure 5d–f suggests that the combined feature set provides a more comprehensive representation of the data.

4.7. Confusion Matrix Analysis

In this section, we analyze the performance of our proposed ensemble model through the confusion matrix, as shown in Table 8. The best performance for each metric is highlighted in bold. The confusion matrix provides a detailed breakdown of the model's predictions, highlighting the number of true positives, true negatives, false positives, and false negatives.

Table 8. Confusion Matrix of the Proposed Ensemble Model.

Confusion Matrix		**Predicted**		
		Benign	**Phishing**	**Recall**
Actual	Benign	5907	50	0.9916
	Phishing	37	1497	0.9759
	Precision	0.9938	0.9677	F1 Score: 0.9718

The confusion matrix reveals several key insights into the performance of our model:

- True Positives (TP): The model correctly identified 1497 phishing instances. This high number of true positives indicates the model's effectiveness in detecting phishing attacks.
- True Negatives (TN): The model correctly classified 5907 benign instances. The high true negative count demonstrates the model's accuracy in identifying legitimate webpages.
- False Positives (FP): There were 50 benign instances incorrectly classified as phishing. Although this number is relatively low, it highlights the importance of further improving the model to minimize false alarms.
- False Negatives (FN): The model incorrectly identified 37 phishing instances as benign. This number, while also low, underscores the need for continuous improvement to ensure that phishing attacks are not missed.

In conclusion, the confusion matrix analysis shows that our ensemble model excels at distinguishing between benign and phishing webpages, with high precision, recall, and F1 scores. These results underscore the model's robustness in enhancing phishing detection capabilities.

4.8. Generalizability Evaluation on Unseen Phishing Data

The primary focus of this study is on accurately identifying phishing websites, with a particular emphasis on minimizing false negatives. To evaluate the generalizability of our ensemble model, we conducted an additional experiment using a completely new phishing dataset that was not part of the original training set. For this experiment, we collected 14,573 phishing URLs and corresponding HTML documents from Mendeley Data. The model, which had been previously trained on our original dataset, was tested on this new phishing data without any further fine-tuning or adjustment of the model's weights.

The model correctly identified 13,919 out of 14,573 phishing instances, resulting in an accuracy of approximately 95.5%. This high accuracy indicates that the model is capable of effectively generalizing to unseen phishing data, maintaining its strong performance even when exposed to phishing tactics that were not included in the training phase. By testing exclusively on phishing data, we focused on assessing the model's robustness in real-world scenarios where detecting phishing attempts is critical. These results suggest that the model is well-suited to generalize across different phishing examples, reinforcing its potential application in diverse phishing detection environments.

Future work could extend this evaluation by incorporating legitimate websites into the test dataset to further validate the model's generalizability across different types of content. To further validate the robustness of our model, future work could involve testing on additional unseen phishing datasets from diverse sources to ensure the model's generalizability across different phishing strategies and tactics.

4.9. Robustness against Adversarial Attacks

In addition to evaluating the model's performance under normal conditions, we conducted experiments to assess its robustness against adversarial attacks. Specifically, we applied the Fast Gradient Sign Method (FGSM) to generate adversarial examples by introducing small perturbations to the input data. These perturbations were designed to test the model's ability to maintain accuracy when faced with adversarially altered inputs.

We tested the model with various levels of perturbation, represented by different epsilon values (ε), ranging from 0 (no perturbation) to 0.1 (significant perturbation). The results of these tests are summarized in Table 9.

Table 9. Adversarial Attack Performance Results (Macro Average).

Epsilon (ε)	Accuracy	Precision	Recall	F1 Score
0	0.9873	0.9875	0.9894	0.9884
0.02	0.9732	0.9466	0.9754	0.9608
0.04	0.9389	0.8841	0.9451	0.9136
0.06	0.8767	0.7994	0.9004	0.8435
0.08	0.7468	0.6981	0.8101	0.7494

As seen from the results, the model's performance begins to degrade as the epsilon value increases. With no perturbation ($\varepsilon = 0$), the model achieves an accuracy of 98.73%, with high macro-averaged precision, recall, and F1-score values. However, even a small perturbation ($\varepsilon = 0.02$) reduces the accuracy to approximately 97.32%, with a noticeable decline in performance, particularly in its ability to correctly classify the benign class.

As the perturbation becomes more significant ($\varepsilon = 0.04$ and above), the model's accuracy drops further. At $\varepsilon = 0.08$, the model's accuracy falls to 74.68%, with the confusion matrix indicating that the model is heavily biased towards misclassifying benign samples as phishing.

These findings highlight the model's vulnerability to adversarial attacks, particularly when small but targeted perturbations are applied. This underscores the need for incorporating more robust defense mechanisms in the model, such as adversarial training or other forms of regularization, to enhance its resilience against such attacks.

4.10. Discussion: Case Analysis

In this section, we analyze specific cases to understand the performance of our proposed model, particularly focusing on instances where the model correctly and incorrectly classified URLs, as detailed in Table 10. This analysis provides insights into the strengths and limitations of our approach.

Table 10. Detailed Case Analysis of Correct and Incorrect URL Classifications by the Proposed Model.

Classification Result	Case	Ground Truth	URL
Correctly classified	(a)	Benign	https://ninecloud.ae/our-services/interior-exterior-paint-contractors-in-dubai/ (accessed on 19 August 2024)
	(b)	Phishing	https://pub-88f64e013ca94e82aa5d15393134722c.r2.dev/logs.html (accessed on 19 August 2024)
Misclassified	(c)	Benign	https://rilm.am/wp-content/uploads/2022/07/12e47ac82164e89a8c15f399384e6572.pdf (accessed on 19 August 2024)
	(d)	Phishing	https://amangroup.co/gy/linkedin_/ (accessed on 19 August 2024)

In Figure 6a, the URL "https://ninecloud.ae/our-services/interior-exterior-paint-contractors-in-dubai/" (accessed on 19 August 2024) was correctly classified as benign. The DOM Graph visualization (Figure 6a) shows a clear and simple structure, which likely contributed to the model's accurate classification. The straightforward and legitimate appearance of the URL, along with the coherent HTML structure, aligns well with benign patterns the model has learned. In Figure 6b, the URL "https://pub-88f64e013ca94e82aa5d15393134722c.r2.dev/logs.html" (accessed on 19 August 2024) was correctly classified as phishing. As depicted in the DOM Graph visualization (Figure 6b), the URL exhibits a complex and suspicious structure. The presence of random characters and a deceptive path indicates phishing characteristics, which the model successfully identified.

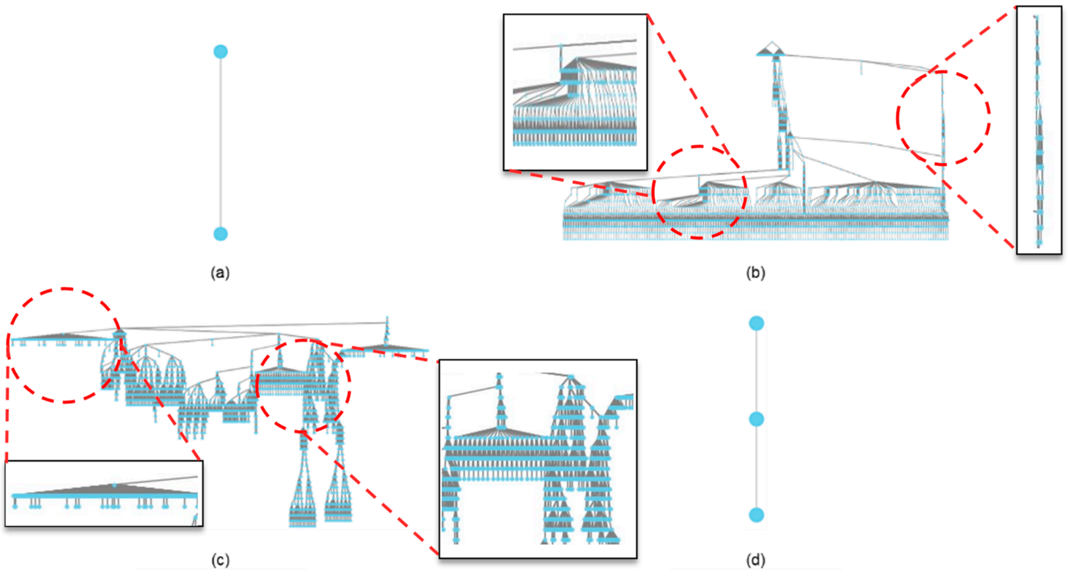

Figure 6. Comparative HTML DOM Graph Visualizations for Benign and Phishing Case Analysis. (**a**) DOM structure of the benign URL "https://ninecloud.ae/our-services/interior-exterior-paint-contractors-in-dubai/ (accessed on 19 August 2024)," showing a clear and simple structure, correctly classified as benign. (**b**) DOM structure of the phishing URL "https://pub-88f64e013ca94e82aa5d15393134722c.r2.dev/logs.html (accessed on 19 August 2024)," showing a complex and suspicious structure, correctly classified as phishing. (**c**) DOM structure of the benign URL "https://rilm.am/wp-content/uploads/2022/07/12e47ac82164e89a8c15f399384-e6572.pdf (accessed on 19 August 2024)," showing a complex structure, incorrectly classified as phishing. (**d**) DOM structure of the phishing URL "https://amangroup.co/gy/linkedin_/ (accessed on 19 August 2024)," showing a relatively simple structure, incorrectly classified as benign.

In Figure 6c, the URL "https://rilm.am/wp-content/uploads/2022/07/12e47ac82164e89a8c15f399384e6572.pdf" (accessed on 19 August 2024) was incorrectly classified as phishing. The DOM Graph visualization (Figure 6c) shows a complex structure that might have misled the model. Despite being a benign URL, its intricate and lengthy format may resemble phishing patterns, leading to a false positive. This highlights the challenge of distinguishing between complex legitimate URLs and phishing URLs. In Figure 6d, the URL "https://amangroup.co/gy/linkedin_/" (accessed on 19 August 2024) was incorrectly classified as benign. The DOM Graph visualization (Figure 6d) shows a relatively simple structure. However, this simplicity might have contributed to the model's failure to recognize it as phishing. The deceptive use of familiar keywords like "linkedin" might have made the URL appear legitimate, resulting in a false negative. This analysis underscores the importance of further refining our model to better distinguish between subtle phishing indicators and legitimate but complex URL structures.

One major issue is that the model only used the DOM structure, which, while improving the model's performance overall, presents a challenge when the DOM structure is too simple or resembles that of a legitimate webpage. Therefore, it is necessary to incorporate additional HTML features, such as HTML DOM tag names and hyperlinks, to improve phishing detection. Additionally, the reliance on static features extracted from URLs and HTML DOM structures may be susceptible to obfuscation by evolving phishing tactics, potentially reducing the model's effectiveness over time. Moreover, some benign URLs with complex structures were misclassified as phishing due to their resemblance to phishing patterns in the HTML DOM. This highlights the need for incorporating contextual information, such as user behavior or dynamic content analysis, to enhance detection accuracy. Moreover, the integration of multi-modal features improved detection rates but also increased the computational complexity of the model, which could be a limitation in real-time applications where processing speed is crucial.

Our findings contribute to the broader field of phishing detection by demonstrating the effectiveness of integrating multi-modal features, such as HTML DOM structures and URL characteristics, to improve detection accuracy. These results suggest that combining different data sources can capture more comprehensive patterns associated with phishing attempts. For future research, exploring the integration of user interaction data and behavioral analytics could provide deeper insights into phishing tactics, offering opportunities to develop more adaptive and robust detection systems. Additionally, investigating the application of real-time analysis techniques and leveraging advances in adversarial learning could further enhance model resilience against sophisticated phishing attacks.

5. Conclusions

In this study, we proposed a phishing detection approach that integrates HTML DOM graph modeling with URL feature analysis using deep learning techniques. By leveraging Graph Convolutional Networks to model HTML DOM graphs and using Convolutional Neural Networks and Transformer Networks to capture character and word sequence features from URLs, our method effectively combines these multi-modal features. This approach addresses the limitations of traditional URL-based phishing detection methods, which often struggle to capture the full context of phishing attacks. Our ensemble model demonstrated significant performance improvements, achieving a 7.03 percentage point increase in classification accuracy compared to state-of-the-art techniques.

The detailed evaluation, including confusion matrix analysis and ablation studies, highlighted the importance of integrating character-based URL features, word-based URL features, and HTML DOM graphs for effective phishing detection. The results validated the superiority of our approach in accurately identifying phishing webpages, due to the complementary strengths of the multi-modal features from URLs and the diverse deep learning models used. Our research makes a significant technical contribution by being the first to combine URL features and HTML DOM graph features and designing a neural network that effectively merges these complementary characteristics.

Despite these improvements, several limitations were identified during the study. Our model relies on static features extracted from URLs and HTML DOM structures, which may be susceptible to obfuscation by evolving phishing tactics, potentially reducing the model's effectiveness over time. Additionally, some benign URLs with complex structures were misclassified as phishing, indicating the need for additional contextual information, such as user behavior or dynamic content analysis, to enhance detection accuracy. While the integration of multi-modal features improved detection rates, it also increased computational complexity, which could be a limitation in real-time applications where processing speed is crucial.

Future research should focus on incorporating additional HTML features, such as DOM tag names and hyperlinks, to provide a more detailed representation of the webpage. Integrating user behavior analysis and browser interaction patterns could offer deeper insights into phishing detection by considering both static properties and dynamic interactions. Additionally, leveraging advancements in adversarial learning to improve robustness against sophisticated phishing tactics is another promising direction. These enhancements can further improve the efficacy of phishing detection systems, making them more comprehensive and reliable.

Author Contributions: Conceptualization, S.-J.B.; formal analysis, J.-H.Y.; funding acquisition, H.-J.K.; investigation, J.-H.Y.; methodology, S.-J.B. and H.-J.K.; visualization, J.-H.Y.; writing—review and editing, S.-J.B. and H.-J.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2021R1F1A1063085).

Data Availability Statement: Data are contained within the article.

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

References

1. Dhamija, R.; Tygar, J.D.; Hearst, M. Why phishing works. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Montreal, QC, Canada, 22–27 April 2006; pp. 581–590.
2. Lee, J.; Wang, J.; de Guzman, M.C.; Gupta, M.; Rao, H.R. Can I Help Prevent Data Breaches in the Workplace? From Routine Activities to Extra-Role Security Behaviors. *IEEE Trans. Technol. Soc.* **2024**. *Early Access*. [CrossRef]
3. Tsai, Y.-D.; Liow, C.; Siang, Y.S.; Lin, S.-D. Toward more generalized malicious url detection models. In Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada, 20–27 February 2024; pp. 21628–21636.
4. Alsharnouby, M.; Alaca, F.; Chiasson, S. Why phishing still works: User strategies for combating phishing attacks. *Int. J. Hum.-Comput. Stud.* **2015**, *82*, 69–82. [CrossRef]
5. Aljofey, A.; Jiang, Q.; Qu, Q.; Huang, M.; Niyigena, J.-P. An effective phishing detection model based on character level convolutional neural network from URL. *Electronics* **2020**, *9*, 1514. [CrossRef]
6. Apruzzese, G.; Subrahmanian, V. Mitigating adversarial gray-box attacks against phishing detectors. *IEEE Trans. Dependable Secur. Comput.* **2022**, *20*, 3753–3769. [CrossRef]
7. Baki, S.; Verma, R.M. Sixteen years of phishing user studies: What have we learned? *IEEE Trans. Dependable Secur. Comput.* **2022**, *20*, 1200–1212. [CrossRef]
8. Tajaddodianfar, F.; Stokes, J.W.; Gururajan, A. Texception: A character/word-level deep learning model for phishing URL detection. In Proceedings of the ICASSP 2020—2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 2857–2861.
9. Vo Quang, M.; Bui Tan Hai, D.; Tran Kim Ngoc, N.; Ngo Duc Hoang, S.; Nguyen Huu, Q.; Phan The, D.; Pham, V.-H. Shark-Eyes: A multimodal fusion framework for multi-view-based phishing website detection. In Proceedings of the 12th International Symposium on Information and Communication Technology, Ho Chi Minh, Vietnam, 7–8 December 2023; pp. 793–800.
10. Yan, X.; Xu, Y.; Cui, B.; Zhang, S.; Guo, T.; Li, C. Learning URL embedding for malicious website detection. *IEEE Trans. Ind. Inform.* **2020**, *16*, 6673–6681. [CrossRef]
11. Gutierrez, C.N.; Kim, T.; Della Corte, R.; Avery, J.; Goldwasser, D.; Cinque, M.; Bagchi, S. Learning from the ones that got away: Detecting new forms of phishing attacks. *IEEE Trans. Dependable Secur. Comput.* **2018**, *15*, 988–1001. [CrossRef]
12. Zhang, H.; Liu, G.; Chow, T.W.; Liu, W. Textual and visual content-based anti-phishing: A Bayesian approach. *IEEE Trans. Neural Netw.* **2011**, *22*, 1532–1546. [CrossRef]
13. Zhu, E.; Chen, Z.; Cui, J.; Zhong, H. MOE/RF: A novel phishing detection model based on revised multiobjective evolution optimization algorithm and random forest. *IEEE Trans. Netw. Serv. Manag.* **2022**, *19*, 4461–4478. [CrossRef]
14. Bozkir, A.S.; Dalgic, F.C.; Aydos, M. GramBeddings: A new neural network for URL based identification of phishing web pages through n-gram embeddings. *Comput. Secur.* **2023**, *124*, 102964. [CrossRef]
15. Nowroozi, E.; Mohammadi, M.; Conti, M. An adversarial attack analysis on malicious advertisement URL detection framework. *IEEE Trans. Netw. Serv. Manag.* **2022**, *20*, 1332–1344. [CrossRef]

16. Sabir, B.; Babar, M.A.; Gaire, R.; Abuadbba, A. Reliability and robustness analysis of machine learning based phishing url detectors. *IEEE Trans. Dependable Secur. Comput.* **2022**. *Early Access*. [CrossRef]
17. Purwanto, R.W.; Pal, A.; Blair, A.; Jha, S. PhishSim: Aiding phishing website detection with a feature-free tool. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 1497–1512. [CrossRef]
18. Çolhak, F.; Ecevit, M.İ.; Uçar, B.E.; Creutzburg, R.; Dağ, H. Phishing Website Detection through Multi-Model Analysis of HTML Content. *arXiv* **2024**, arXiv:2401.04820.
19. 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]
20. Prasad, A.; Chandra, S. PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning. *Comput. Secur.* **2024**, *136*, 103545. [CrossRef]
21. Bu, S.-J.; Kim, H.-J. Optimized URL feature selection based on genetic-algorithm-embedded deep learning for phishing website detection. *Electronics* **2022**, *11*, 1090. [CrossRef]
22. Wang, M.; Li, H.; Tao, D.; Lu, K.; Wu, X. Multimodal graph-based reranking for web image search. *IEEE Trans. Image Process.* **2012**, *21*, 4649–4661. [CrossRef]
23. Chartsias, A.; Joyce, T.; Giuffrida, M.V.; Tsaftaris, S.A. Multimodal MR synthesis via modality-invariant latent representation. *IEEE Trans. Med. Imaging* **2017**, *37*, 803–814.
24. Fu, Y.; Hospedales, T.M.; Xiang, T.; Gong, S. Learning multimodal latent attributes. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *36*, 303–316.
25. Liu, K.; Xue, F.; Guo, D.; Sun, P.; Qian, S.; Hong, R. Multimodal graph contrastive learning for multimedia-based recommendation. *IEEE Trans. Multimed.* **2023**, *25*, 9343–9355. [CrossRef]
26. Pang, L.; Zhu, S.; Ngo, C.-W. Deep multimodal learning for affective analysis and retrieval. *IEEE Trans. Multimed.* **2015**, *17*, 2008–2020. [CrossRef]
27. Wang, D.; Cui, P.; Ou, M.; Zhu, W. Learning compact hash codes for multimodal representations using orthogonal deep structure. *IEEE Trans. Multimed.* **2015**, *17*, 1404–1416. [CrossRef]
28. Wu, F.; Lu, X.; Song, J.; Yan, S.; Zhang, Z.M.; Rui, Y.; Zhuang, Y. Learning of multimodal representations with random walks on the click graph. *IEEE Trans. Image Process.* **2015**, *25*, 630–642. [CrossRef] [PubMed]
29. Zheng, W.-L.; Liu, W.; Lu, Y.; Lu, B.-L.; Cichocki, A. Emotionmeter: A multimodal framework for recognizing human emotions. *IEEE Trans. Cybern.* **2018**, *49*, 1110–1122. [CrossRef]
30. Ding, C.; Tao, D. Robust face recognition via multimodal deep face representation. *IEEE Trans. Multimed.* **2015**, *17*, 2049–2058. [CrossRef]
31. Li, W.; Joo, J.; Qi, H.; Zhu, S.-C. Joint image-text news topic detection and tracking by multimodal topic and-or graph. *IEEE Trans. Multimed.* **2016**, *19*, 367–381. [CrossRef]
32. Monwar, M.M.; Gavrilova, M.L. Multimodal biometric system using rank-level fusion approach. *IEEE Trans. Syst. Man Cybern. Part B* **2009**, *39*, 867–878. [CrossRef] [PubMed]
33. Tang, S.; Guo, D.; Hong, R.; Wang, M. Graph-based multimodal sequential embedding for sign language translation. *IEEE Trans. Multimed.* **2021**, *24*, 4433–4445. [CrossRef]
34. Zhang, Z.; Ma, J.; Du, J.; Wang, L.; Zhang, J. Multimodal pre-training based on graph attention network for document understanding. *IEEE Trans. Multimed.* **2022**, *25*, 6743–6755. [CrossRef]
35. Ariyadasa, S.; Fernando, S.; Fernando, S. Combining long-term recurrent convolutional and graph convolutional networks to detect phishing sites using URL and HTML. *IEEE Access* **2022**, *10*, 82355–82375. [CrossRef]
36. Le, H.; Pham, Q.; Sahoo, D.; Hoi, S.C. URLNet: Learning a URL representation with deep learning for malicious URL detection. *arXiv* **2018**, arXiv:1802.03162.