UTM UNIVERSITI TEKNOLOGI MALAYSIA SCHOOL OF COMPUTING Faculty of Engineering

DeepPhish-X: Multi-Modal Feature Engineering for Phishing Detection Using Hybrid Models of Computer Vision, Natural Language Processing, and Graph Neural Networks

Program Name:	**Masters of Science (Data Science)**
Subject Name:	**Project 1 (MCST1043)**
Student Name:	Cui ZhiWen
Metric Number:	MCS241040
Student Email & Phone:	cuizhiwen@graduate.utm.my
Project Title:	DeepPhish-X: Multi-Modal Feature Engineering for Phishing Detection Using Hybrid Models of Computer Vision, Natural Language Processing, and Graph Neural Networks
Supervisor 1:	
Supervisor 2 / Industry Advisor(if any):	

This paper introduces **DeepPhishX**, a multimodal deep learning system specifically designed to detect phishing web pages, a task that is critical in the field of cybersecurity and data protection. Traditional methods rely mainly on URL feature analysis, but they cannot effectively identify the multi-layered logic of advanced phishing attacks. To this end, DeepPhishX innovatively combines URL feature analysis with HTML DOM graph structure modeling, and uses a variety of model fusion methods such as **Graph Convolutional Network (GCN)**, **Character-level Convolutional Neural Network (CNN)**, and **Word Sequence Transformer** to achieve in-depth analysis of the structure and content of phishing web pages.

We rigorously evaluated DeepPhishX on a dataset that covers real-world web page data (HTML DOM graphs and URLs, totaling more than 80 million nodes/edges) from 2012 to 2024. The experimental results show that compared with the current state-of-the-art methods, DeepPhishX improves the classification accuracy by **7.03 percentage points**, and the ablation experiment also shows that each feature module has a significant contribution to the performance. These results verify the effectiveness of hybrid deep learning in combining DOM structure and URL features, and prove that DeepPhishX can provide a more accurate and comprehensive solution for identifying malicious web pages.

1. Introduction

The rapid development of the Internet has changed the way people interact with each other, and has also brought about increasingly serious cybersecurity threats, and **phishing attacks** have become one of the most common and harmful forms of online fraud. Phishing sites trick users into entering sensitive information by pretending to be legitimate, and these cloaking methods are becoming more and more subtle, making traditional basic URL detection methods strained.

While URLs can be easily imitated, phishing pages are difficult to match legitimate websites in terms of **HTML DOM structure**. Structural nuances, often overlooked by attackers, are the key to effective defense. Therefore, it is not enough to just look at the URL; only an in-depth analysis of the logic of the web page structure can reveal the true intent of the phishing website.

To address this challenge, we propose a multimodal fusion strategy:

- **Character-level CNNs**: Automatically identify character patterns (e.g., typos, abnormal combinations) in URLs.

- **Lexical Sequence Transformer**: Captures part-of-speech semantics in URLs and models contextual relationships (e.g., spoofing intent in "login-update").
- **HTML DOM GCN**: Models the structure of the page, characterizes tags, hierarchies, and structural features, and identifies different structural fingerprints of normal and phishing pages.

We propose a new idea of structurally modeling phishing pages: focusing on the **structure of the DOM graph** to supplement the lack of URL features. A multi-expert system integrating CNN, Transformer, and GCN was constructed, and the feature advantages were fused through the attention mechanism. The effectiveness of the proposed method is verified on large-scale real-world datasets, and the accuracy is significantly improved compared with existing technology.

2. Related Work

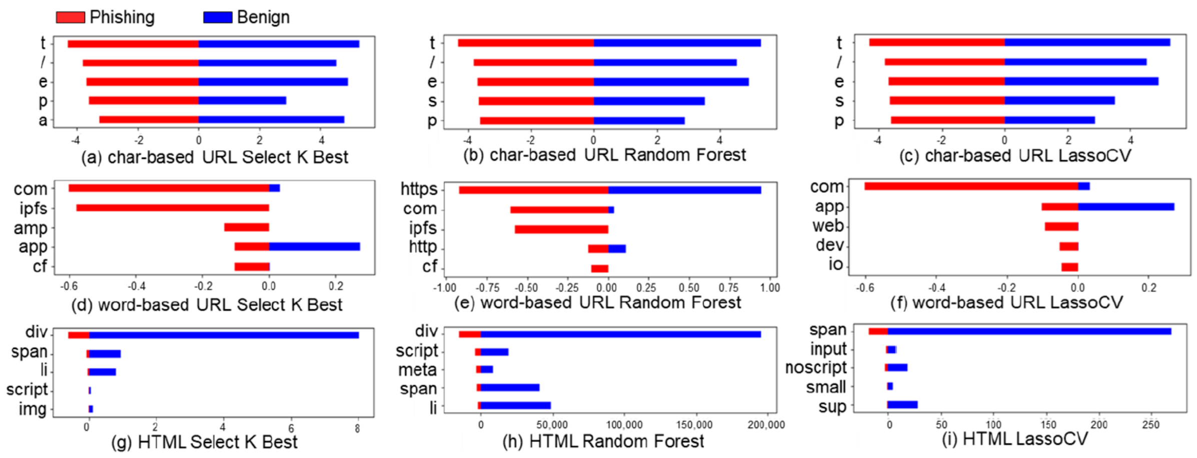
Phishing detection has long been a key area of cybersecurity research. As described in Table 2, various methods have been developed over the years to address this problem.

URL-based Approach

Early approaches focused primarily on analyzing URL characteristics, taking into account factors such as URL length, suspicious substrings, and domain reputation. For example, the **Texception model** uses convolutional layers to analyze character-level and word-level information for URLs, achieving significant performance on large datasets. Advances in phishing detection have seen the integration of multiple machine learning techniques, such as **MOE/RF**, which combines multi-objective evolutionary optimization with random forests to produce high accuracy and recall. Similarly, **GramBeddings** uses a four-channel architecture that combines CNNs, LSTMs, and attention layers to demonstrate significant accuracy on a variety of datasets. The use of adversarial examples, such as **URLBUG**, highlights the challenges posed by adversarial attacks that can degrade the performance of machine learning models. Notably, URLBUG performs lower than other models because it tests on adversarial URLs generated by deliberately circumventing detection, demonstrating the difficulty of maintaining robustness when processing generated data. For example, one approach combines multiple machine learning techniques to analyze the lexical characteristics of URLs and web crawled content, integrating URL structure and web content for a more comprehensive detection approach. Another approach explores embedding URL components and testing adversarial attacks, enhancing the robustness of the model and making it more effective in responding to complex evasion techniques.

HTML-based Approach

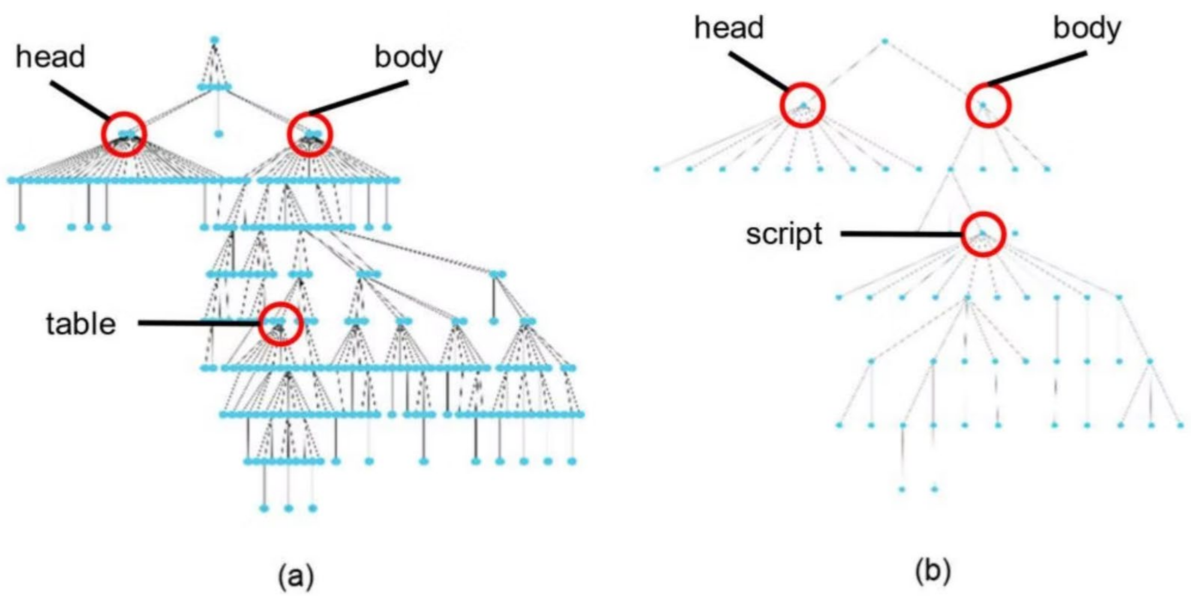
While URL-based methods provide valuable insights, they often fail to capture the full context of phishing attacks. HTML-based methods, such as **PhishSim**, address this limitation by analyzing the content and structure of web pages, achieving high detection rates. Recent research has increasingly focused on integrating URL and HTML features for a more comprehensive detection strategy. For example, one approach combines MLPs for structured data with NLP models for HTML content, fusing embeddings to improve detection accuracy.



Multimodal Methods

The **WebPhish framework** combines raw URL and HTML content analysis, achieving high accuracy, and demonstrating the effectiveness of multimodal methods. The **PhiUSIIL framework** leverages URL similarity indexing and incremental learning to adapt to real-time threats, achieving near-perfect accuracy, precision, and recall, which highlights the potential of real-time adaptive models in phishing detection. Another notable method combines raw URL, HTML tags, and image analysis, using word embeddings and convolutional layers, achieving high accuracy, and demonstrating the advantages of combining multiple data types for phishing detection. The evolution of related research, from simple URL feature analysis to complex HTML and multimodal integration, directly reflects the escalating sophistication of phishing attacks. Initially, simple URL checks might have been sufficient, but as attackers learned to mimic URLs and employ more complex web page structures, detection methods had to adapt accordingly. The emergence of adversarial examples and the decline in performance of the URLBUG model further emphasize the continuous ″arms race″ between attackers and defenders. This indicates that cybersecurity research is inherently reactive and adaptive. Developing increasingly sophisticated deep learning models (such as those utilizing GCNs and Transformers) is a necessary response to the adaptable and innovative nature of cybercriminals. This implies a need for sustained acceleration in research and development to stay ahead of new attack vectors, obfuscation techniques, and adversarial maneuvers, emphasizing that no single, static defense mechanism can remain effective indefinitely.

DeepPhish-X optimizes feature representation for phishing detection by effectively modeling the complex dependencies between HTML tags within the DOM structure using **Graph Convolutional Networks**—a graph neural network technique—thus distinguishing it from existing research. Furthermore, DeepPhish-X incorporates **Transformer networks**—a natural language processing model—to combine URL features with HTML DOM graph features, enabling the model to selectively focus on and extract complementary relationships between these multimodal features. This "complementary" nature implies that these disparate pieces of information, when fused, provide a richer and more robust understanding of the true nature of a web page than any single modality alone. This precise feature integration enhances overall detection accuracy and robustness. This principle of synergy—where the combined intelligence of different data types and specialized processing models yields superior results—is critical in designing effective AI systems in complex adversarial domains like cybersecurity.



Method	Representation	Description	Dataset	Performance
Multimodal Classification	Raw URL, HTML content	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	Combines raw URL, 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 Classification	Character-level, Word-level	Analyzes character and word-level URL information using parallel convolutional layers	1.7 million samples collected from Microsoft Browse telemetry data	TPR: 47.95% Error Rate: 0.28%
URL Features	Combines multi-objective evolutionary optimization	Five different URL datasets (Kaggle,	Accuracy : 99.04% Recall: 99.48%	

Method	Representation	Description	Dataset	Performance
	with random forests for phishing detection	Mendeley Data)		
N-gram Embeddings	Utilizes N-gram embeddings and a four-channel architecture, including CNN, LSTM, attention layers	Alexa, Majestic (benign): 400K Phishtank, Openphish (phishing): 400K	Accuracy: 98.27% F1 Score: 98.26%	
Adversarial URL Generation	A method for generating adversarial URLs by obfuscating the domain, path, and TLD parts of URLs to test the robustness of machine learning-based phishing URL detectors	Dataset contains a total of 193,386 samples from five different sources Benign: 96,693 Phishing: 96,693	(Performance degraded due to testing on generated adversarial data) Accuracy: -41.62% F1 Score: -59.17%	
Lexical, Web Crawling	Combines multiple machine learning	Dataset contains a total of 3,980,870	Accuracy: 99.63% Precision: 99.60%	

Method	Representation	Description	Dataset	Performance
	techniques to extract and analyze lexical and web crawled features	samples from 12 different sources		
URL Embedding	Analyzes the robustness of machine learning models against adversarial attacks	Alexa (benign): 10,000 Phishtank (phishing): Not specified	Precision: 91% Recall: approx. 93% F1 Score: approx. 92.5%	
HTML Classification	HTML Content	Compares HTML content with known phishing pages using normalized compression distance	Common Crawl (benign): 180,302 Phishtank (phishing): 9034	AUC: 98.68% TPR: approx. 90% FPR: 0.58%
HTML Content	Integrates MLP for structured data with NLP models for HTML content, fusing embeddings	Alexa (benign): 2000 Openphish (phishing): 2000	Accuracy: 97.18% F1 Score: 96.80%	

Method	Representation	Description	Dataset	Performance
URL, HTML	Proposes PhiUSIIL, a phishing URL detection framework that combines URL similarity indexing and incremental learning to adapt to real-time threats	Open PageRank Initiative Anon (benign): 134,850 Phishtank, OpenPhish, Malware World (phishing): 100,945	Accuracy: 99.97% Precision: 99.97% Recall: 99.98% F1 Score: 99.98%	

3. Proposed Method

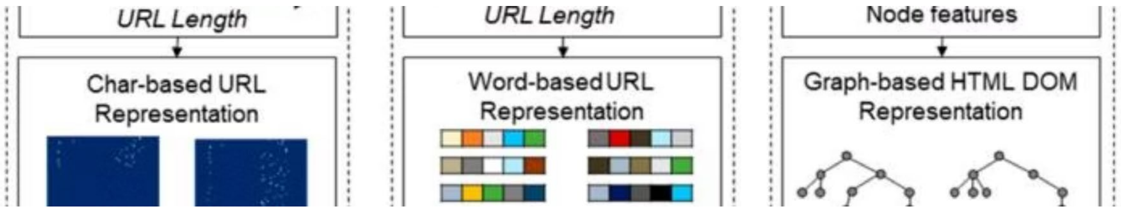
This section introduces **DeepPhish-X**, a phishing detection framework that utilizes Graph Convolutional Networks and Transformer Networks, integrating multimodal features from HTML DOM graphs and URL features. DeepPhish-X combines different deep learning models to achieve more accurate phishing web page classification. DeepPhish-X aims to integrate various features to detect phishing attacks, while recognizing that features such as URLs, SSL certificates, HTML DOM, web page content, HTML headers, and protocols, although available, are often susceptible to manipulation. The core idea is that the most influential feature in determining whether a case is phishing varies from instance to instance. Therefore, it is crucial to leverage as many features as possible while excluding those that might hinder learning due to manipulation, thereby maximizing phishing detection capabilities.

3.1. URL and HTML Data Representation for Phishing Detection

To effectively detect phishing web pages, DeepPhish-X employs multimodal feature engineering, analyzing and representing data from multiple perspectives. This section outlines the methods used to represent URL and HTML data, which are crucial for identifying patterns and characteristics unique to phishing attempts. DeepPhish-X uses deep learning techniques to process and extract features from both URL and HTML content, providing a robust framework for phishing detection.

3.1.1. Character-based URL Feature Extraction (Inspired by Computer Vision)

Character-based URL feature extraction in DeepPhish-X involves converting each URL into a matrix representation using **one-hot encoding**. Each character in the URL is mapped to its corresponding ASCII value, yielding a numerical representation. The ASCII values are then padded or truncated to a fixed length of 128 characters to maintain consistency across all URLs, and one-hot encoded into a 128x128 matrix. This fixed-size input is suitable for CNN processing, borrowing techniques from computer vision for handling grid-like data.


URL Length
Char-based URL Representation
URL Length
Word-based URL Representation
Node features
Graph-based HTML DOM Representation

Algorithm 1: Character-based URL Representation for Phishing Detection

```
Input: URL string

Output: 128x128 matrix representing the URL

Function ConvertToASCII(url):

Return [ord(char) for char in url]

Function ApplyPadding(ascii_values):

max_length = 128

Return (ascii_values[:max_length] + [0] * (max_length -
len(ascii_values)))[:max_length]

Function OneHotEncode(ascii_values):

matrix = zero matrix of size 128x128
```

```
For i, value in enumerate(ascii_values) do:

matrix[i][value] = 1

Return matrix

ascii_values = ConvertToASCII(URL)

padded_values = ApplyPadding(ascii_values)

matrix = OneHotEncode(padded_values)
```

Given this representation, DeepPhish-X utilizes **CNNs** for feature extraction. CNNs are a class of deep neural networks specifically designed to process data with a grid-like topology, such as images or character sequences. They are particularly effective in tasks involving spatial hierarchies and pattern recognition. In the context of character-based URL feature extraction, CNNs are employed to capture local patterns and dependencies between characters in the one-hot encoded URL matrix. By applying convolutional filters across the matrix, CNNs can identify and learn significant character combinations and sequences that may indicate a phishing attempt. The convolution operation is given by:

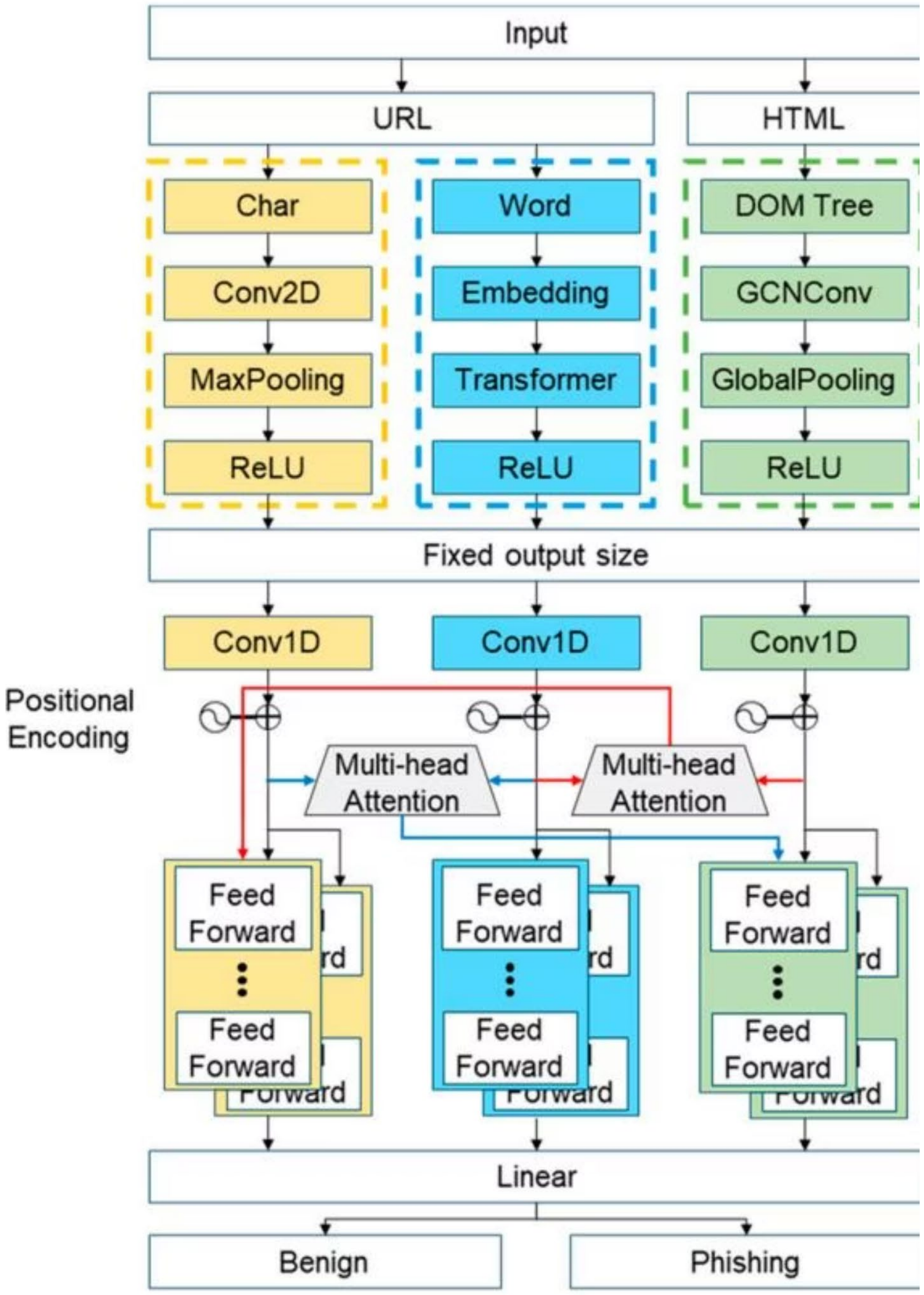
$$H_l=f(\phi(W_l,X)+b_l)$$

where H_l is the output feature map of layer l, W_l represents the convolutional filter weights, ϕ denotes the convolution operation, b_l is the bias term, and $f(\cdot)$ is the activation function. The primary advantage of CNNs in this application lies in their ability to automatically learn and detect spatially invariant patterns, which makes them well-suited for identifying phishing URLs where certain character sequences or patterns may recur across different URLs.

To train the CNN component of DeepPhish-X, the loss function is defined as the **categorical cross-entropy loss**, which measures the discrepancy between predicted probabilities and true labels. The loss function L_{CNN} is given by:

$$L_{CNN}=-\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}y_{i,c}\log(\sigma(W_O H_L(X_i)+b_O)_c)$$

where N is the number of URL samples, C is the number of categories, $y_{i,c}$ is the true label for the i-th sample in category c, X_i is the i-th input sample, $H_L(X_i)$ is the feature map of the i-th input character-level URL obtained from the last layer LCNN of the CNN, WO and bO are the weight and bias of the output layer, respectively, σ is the softmax function, and pi,c is the predicted probability that the i-th input belongs to category c, calculated as σ (WOHL(Xi)+bO)c.


Input
URL
HTML
Char
Conv2D
MaxPooling
ReLU
Word
Embedding
Transformer
ReLU
DOM Tree
GCNConv
GlobalPooling
ReLU
Fixed output size
Conv1D
Conv1D
Conv1D
Positional Encoding
Multi-head Attention
Multi-head Attention
Feed Forward
Feed Forward
Feed Forward
Feed Forward
Feed Forward
Feed Forward
Linear
Benign
Phishing

Padding or truncating URLs to a fixed length of 128 characters and converting them into a 128x128 one-hot encoded matrix is a common yet critical preprocessing step in deep learning for handling variable-length inputs. This highlights the practical engineering challenges faced when applying deep learning to inherently variable-length data such as URLs. This design choice implies a trade-off: while it enables efficient batch processing and model training, it can also lead to potential loss of certain unique features due to truncation in very long URLs or dilution of patterns due to padding in very short ones. This might impact the model's performance on extreme URL lengths. This suggests that max_length (128 in this case) is a hyperparameter that balances capturing sufficient information with computational efficiency and memory constraints, and its optimization is crucial for robust performance across varying URL lengths.

3.1.2. Word-based URL Feature Extraction (Natural Language Processing)

Word-based URL feature extraction in DeepPhish-X begins by **tokenizing URLs** using special characters such as slashes (/), dots (.), and hyphens (-) as delimiters. These tokens are further refined by filtering out common stopwords and retaining only semantically meaningful words. Subsequently, the tokens undergo frequency analysis to construct a vocabulary dictionary comprising the 5000 most frequently occurring words, each mapped to a unique integer identifier. The tokenized URLs are converted into sequences of integers and padded to a uniform length of 20 tokens to facilitate batch processing. These integer sequences serve as inputs to a **Transformer network**, a key component in natural language processing, which consists of an embedding layer, a multi-head attention mechanism with two attention heads, and a position-wise feed-forward network. The attention mechanism computes context relevance by evaluating the significance of each token in the sequence, allowing the model to effectively capture long-range dependencies.

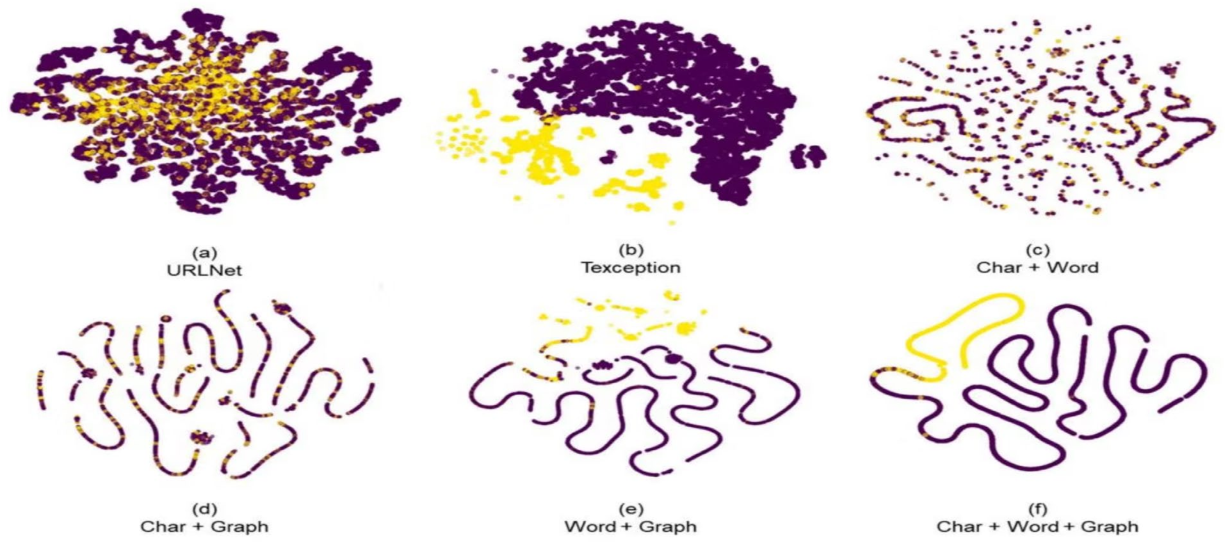
Algorithm 2: Word-based URL Representation for Phishing Detection

```
Input: List of URLs

Output: URLs transformed into integer sequences

Initialize empty dictionary D

Define special characters S = ″!@#%^&*()+-={}[],./<>?″
```

```
Function SplitURL(url):
Return split(url, S)
Function BuildDictionary(words):
Count and sort words by frequency
Return top 5000 words with indices 1 to 5,000
Function TransformURL(url, D):
words = SplitURL(url)
sequence = [D[word] if word in D else 0 for word in words]
Return (sequence[:20] + [0] * 20)[:20]
all_words = flatten (SplitURL(url) for url in list_of_URL)
D = BuildDictionary(all_words)
all_sequences = [TransformURL(url, D) for url in list_of_URL]
```

To effectively represent URLs as sequences of words, DeepPhish-X first segments each URL string using a predefined set of special characters. This segmentation process transforms the URL into a list of words. Next, a dictionary containing the most frequently occurring words in the dataset is constructed, mapping each word to a unique integer. The URLs are then converted into these integer sequences, which the Transformer network can process to capture contextual dependencies between words. Algorithm 2 illustrates the detailed steps for converting URLs into word-based integer sequences.

Following this preprocessing, the transformed URL sequences are fed into a Transformer network to capture contextual relationships and dependencies among words. Transformer networks are particularly well-suited for this task due to their proficiency in modeling long-range dependencies and capturing intricate patterns within sequential data, a hallmark of natural language processing. In the context of word-based URL feature extraction, DeepPhish-X leverages the Transformer's ability to understand the contextual meaning of words in a URL by employing a **self-attention mechanism**. The attention mechanism, as shown, computes the relevance of different parts of a sequence, allowing the model to dynamically weigh the importance of each token:



$$Attention(Q,K,V)=softmax(QKT/dk)V$$

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

For a given input sequence X, multi-head attention is defined as:

$$MultiHead(Q,K,V)=Concat(head1,...,headh)WO$$

$$headi=Attention(QWiQ,KWiK,VWiV)$$

where WiQ, WiK, WiV are learned projection matrices. The output of the multi-head attention is subsequently passed through a feed-forward neural network.

Let x be the output of the multi-head attention. The feed-forward neural network is defined as:

$$FFN(x)=max(0,xW1+b1)W2+b2$$

where W1, W2, b1, b2 are learned parameters. The final output of the Transformer model is given by Z=Transformer(X), representing the high-level features extracted from the input sequence. The Transformer function is defined as:

$Transformer(X)=FFN(MultiHead(Q,K,V))$

These high-level features extracted by the Transformer are then fed into DeepPhish-X's final classification layer to detect phishing web pages. To train the Transformer component, the loss function is defined as the **categorical cross-entropy loss**, which measures the discrepancy between predicted probabilities and true labels. The loss function LTransformer is given by:

$LTransformer=-N1\sum i=1N\sum c=1Cyi,clog(\sigma(WOZi+bO)c)$

where Xi denotes the i-th input sequence, and Zi is the Transformer for the i-th input sequence. WO and bO are the weight and bias of the output layer, respectively, σ is the softmax function, and pi,c is the predicted probability that the i-th input belongs to category c, calculated as σ(WOZi +bO)c.

DeepPhish-X employs both **character-based (CNN, inspired by computer vision)** and **word-based (Transformer, from natural language processing)** URL analysis. Character-level analysis excels at detecting subtle typos, character insertions, or anomalous character sequences (e.g., "gooogle.com" or encoded characters), which are common phishing tactics. Conversely, word-based analysis focuses on the semantic meaning of words and their contextual relationships (e.g., "login.php" appearing in an unusual domain, or combinations like "secure-update-account"). This dual approach ensures that DeepPhish-X can identify phishing attempts that leverage different linguistic granularities. This suggests that phishing attacks are multifaceted, exploiting both low-level typographical errors and high-level deceptive narratives. A comprehensive detector needs to capture both. The combination of these two modalities provides a richer and more robust understanding of a URL's malicious intent, making the entire DeepPhish-X system more resilient to various forms of URL obfuscation and mimicry.

The choice of **Transformer networks**, specifically their multi-head attention mechanism, for word-based URL feature extraction is significant because they "excel at capturing long-range dependencies" and "contextual relevance." Unlike simpler methods that might just look for suspicious keywords, Transformers can understand the relationships between words in a URL and dynamically weigh their importance within the sequence. For instance, "secure-login.com" might be benign, but "login.secure-update.com" could be suspicious. The order and relationship of words are crucial. This indicates that DeepPhish-X aims to go beyond simple lexical feature matching. By understanding the contextual meaning of words within a URL, the Transformer can identify more subtle malicious intent clues. This capability is vital for detecting sophisticated phishing URLs that might use legitimate words in a deceptive order or combination, making the model more intelligent in interpreting URL semantics.

3.1.3. DOM Graph Feature Extraction (Graph Neural Networks)

DeepPhish-X uses **BeautifulSoup 4.12.3** (a Python 3.9 library for web scraping) to parse HTML documents to extract the **DOM tree structure**. Each HTML element (e.g., `<div>`, `<a>`, `<p>`) is represented as a node, and edges are established based on parent-child relationships within the DOM hierarchy. This graph representation is a fundamental element of graph neural networks, which captures element types and their relational context, crucial for understanding structural nuances in phishing pages. The HTML content of each web page is first parsed into a DOM tree using BeautifulSoup. The hierarchical relationships of these elements are preserved, with parent-child relationships represented as edges in the graph. Each node in the graph represents an HTML tag, and these tags' attributes (such as `id`, `class`, and `href` attributes) were initially considered as potential features. However, to maintain computational efficiency and focus on the structural aspects, node features are limited to tag names and simplified attribute representations, such as the presence or absence of key attributes (e.g., `href`, `src`).

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

```
Input: HTML document

Output: Graph representation of the HTML DOM tree

Function ParseHTML(html_document):

dom_tree = Parse html_document into a DOM tree

Return dom_tree

Function BuildGraph(dom_tree):

If dom_tree is empty Then Return 0 # No nodes or edges present
```

```
Initialize graph as an empty graph
Use a queue to perform level-order traversal of dom_tree
While queue is not empty Do:
node = Dequeue()
Add node to graph as a vertex
For each child of node Do:
Add child to graph as a vertex
Add an edge from node to child in the graph
End For
End While
Return graph
dom_tree = ParseHTML(HTML document)
graph = BuildGraph(dom_tree)
```

This process begins by parsing the HTML document to construct a DOM tree. This tree structure is then transformed into a graph, where nodes represent HTML elements and edges represent parent-child relationships between these elements. Algorithm 3 provides this transformation's detailed steps. Once the DOM tree is represented as a graph, DeepPhish-X proceeds to use **Graph Convolutional Networks (GCNs)**—a key component of graph neural networks—for feature extraction. GCNs are well-suited for this task as they excel at capturing the relational and structural information inherent in graph data. The following formulas outline the operations involved in applying GCNs to the graph representation of HTML DOM graphs.

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 convolution operation is given by:

$$H(l+1)=ReLU(\hat{D}-21\hat{A}\hat{D}-21H(l)W(l))$$

where $\hat{A}=A+I$ is the adjacency matrix with self-connections, $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$, $H(l)$ is the activation matrix of the l-th layer, $W(l)$ is the weight matrix of the l-th GCN layer, and ReLU is the activation function applied element-wise. The initial input $H(0)$ of the GCN is the feature matrix X derived from the node attributes. The output of the final GCN layer $H(L)$ is used as the high-level feature representation of the graph.

To train DeepPhish-X's GCN component, the **categorical cross-entropy loss function** is used, which measures the discrepancy between predicted class probabilities and true labels. The loss function LGCN is defined as:

$$LGCN=-N1\sum i=1N\sum c=1Cyi,c\log(\sigma(W0H(L)(Xi)+b0)c)$$

where N is the number of URL samples, C is the number of categories, yi,c is the true label for the i-th sample in category c, σ is the softmax function, Xi is the i-th input graph sample, and $H(L)(Xi)$ is the output from the last layer of the GCN for the i-th input graph. The use of GCNs significantly improves the accuracy of phishing detection by effectively leveraging the structural information inherent in HTML DOM graphs.

DeepPhish-X's core premise is that while phishing sites might mimic legitimate URLs, they rarely perfectly replicate the HTML structure. By transforming this structure into a graph and using GCNs to process it, the model essentially learns the web page's **"structural fingerprint."** Legitimate websites often have well-organized, complex, and consistent DOM structures, while phishing websites (often hastily created or templated) may exhibit simpler, irregular, or anomalous structural patterns. This underlying architectural difference is harder for attackers to perfectly imitate or obfuscate than superficial URL changes. This suggests that how a web page is constructed—its underlying architecture and the relationships between its elements—provides a more robust and less forgeable indicator of malicious intent than its superficial appearance or basic content. This moves phishing detection beyond lexical or content-based analysis into an examination of structural integrity, representing a significant advancement in detecting complex and evasive phishing attempts.

HTML DOM trees are inherently hierarchical and graph-like, not perfectly conforming to traditional grid-like (suitable for CNNs) or purely sequential (suitable for RNNs/Transformers) data structures. The explicit choice of **Graph Convolutional Networks (GCNs)** is made because they "excel at capturing the relational and structural information inherent in graph data." This highlights an intentional and intelligent selection of deep learning architecture specifically tailored to the intrinsic topological structure of the data. This suggests an increasing sophistication in applying deep learning to cybersecurity problems. This approach allows DeepPhish-X to leverage rich relational information within the HTML DOM that might be lost or difficult to capture with less suitable architectures, leading to improved detection accuracy.

3.2. Integrated Classifier Leveraging Character-based URL, Word-based URL, and HTML DOM Graph Features (Hybrid Model)

This section presents **DeepPhish-X**, a multimodal ensemble model for phishing detection. Various features can be leveraged for phishing detection, such as URL, SSL certificates, HTML DOM, web page content, HTML headers, and protocols, among others. However, many of these features can be manipulated to appear legitimate, thereby potentially deceiving both users and phishing detection systems. The key lies in recognizing that the most influential feature in determining whether a case is phishing varies for each instance. Therefore, it is crucial to leverage as many features as possible while excluding those that might hinder learning due to manipulation, thereby maximizing phishing detection capabilities.

DeepPhish-X introduces an **ensemble classification model** that integrates the strengths of CNNs (for computer vision-inspired character-based URL features), Transformers (for natural language processing-based word-based URL features), and GCNs (for graph neural network-based HTML DOM features). This multimodal approach aims to capitalize on the complementary nature of these diverse feature representations to enhance the overall accuracy and robustness of phishing detection. The architecture of this hybrid ensemble model is depicted in Figure 4.

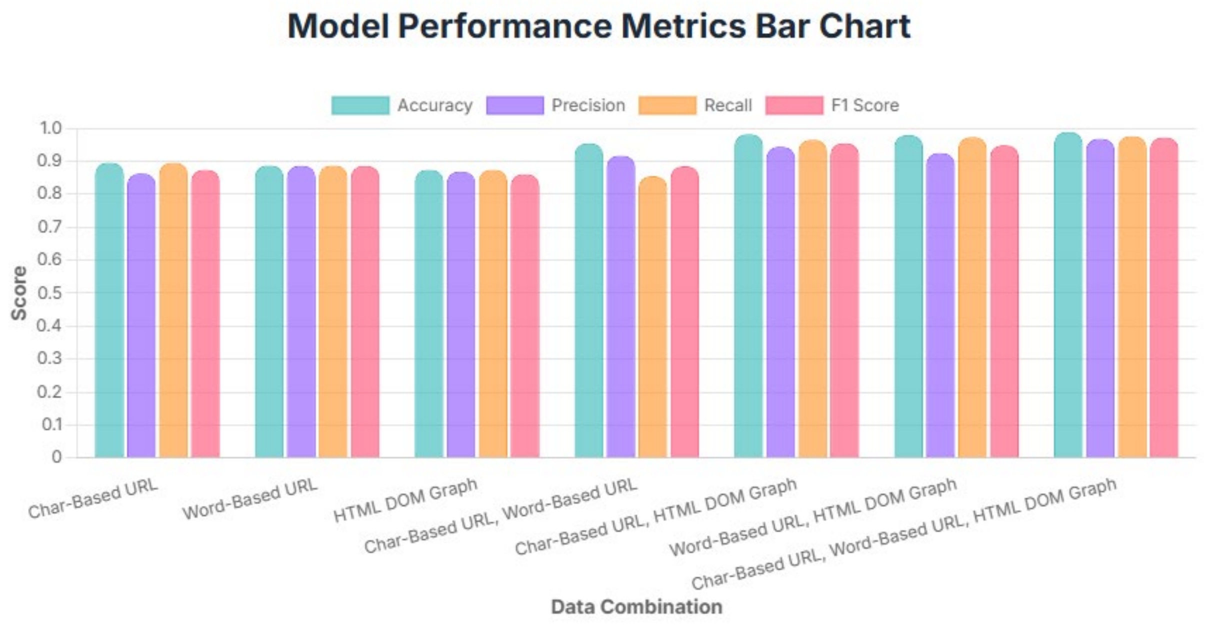
The architecture of DeepPhish-X is designed to effectively integrate multiple data modalities for phishing detection, combining URL and HTML DOM features. The selection of model components and hyperparameters followed empirical experimentation and best practices in the field of deep learning. **Convolutional Neural Networks (CNNs)** are employed to process character-based

URL features, leveraging their strength in capturing local spatial hierarchies, a technique common in computer vision. The architecture includes two convolutional layers, with kernel size 3x3, chosen for their ability to detect small patterns and variations in character sequences. ReLU activation functions are used to introduce non-linearity, and max-pooling layers are included to reduce dimensionality and computational cost. DeepPhish-X utilizes **Transformer networks** for word-based URL analysis because they demonstrate superior capabilities in modeling long-range dependencies and context within sequences, a key strength in natural language processing.

4.1. Dataset

To address the significant class imbalance present in the benign dataset, which could negatively impact the evaluation of experimental results, DeepPhish-X undertook **careful downsampling**. Specifically, the benign dataset was reduced to 60,000 instances to achieve a more balanced dataset, ensuring a fair comparison with phishing data. This step was crucial for maintaining the validity of the evaluation and avoiding skewed results that might misrepresent the model's performance. A total of 38,060 phishing instances were collected and carefully selected to cover a wide range of phishing techniques. The HTML content for 14,912 instances was successfully retrieved by parsing the HTML content with BeautifulSoup. This successfully parsed subset of phishing data provides a strong foundation for further analysis and modeling, ensuring experiments are based on high-quality data.

In an effort to give back to the research community and further support advancements in phishing detection, this study plans to release this dataset as an **open-source benchmark**. This dataset, comprising diverse and comprehensive phishing and benign data, will serve as a vital resource for effectively benchmarking future phishing detection models. The objective of this study is to foster innovation and drive progress in this critical area of cybersecurity. The explicit statement that "To address the significant class imbalance present in the benign dataset… we undertook careful downsampling… to achieve a more balanced dataset, ensuring a fair comparison" is a crucial methodological decision. If left unaddressed, highly imbalanced datasets can lead to models that appear accurate but are actually biased towards the majority class, performing poorly on the minority class (phishing in this context), which is often the critical one to detect correctly. Downsampling ensures that DeepPhish-X is not misled by the imbalanced dataset, leading to more reliable and generalizable performance metrics. This highlights the importance of data preprocessing and ethical data handling in machine learning.

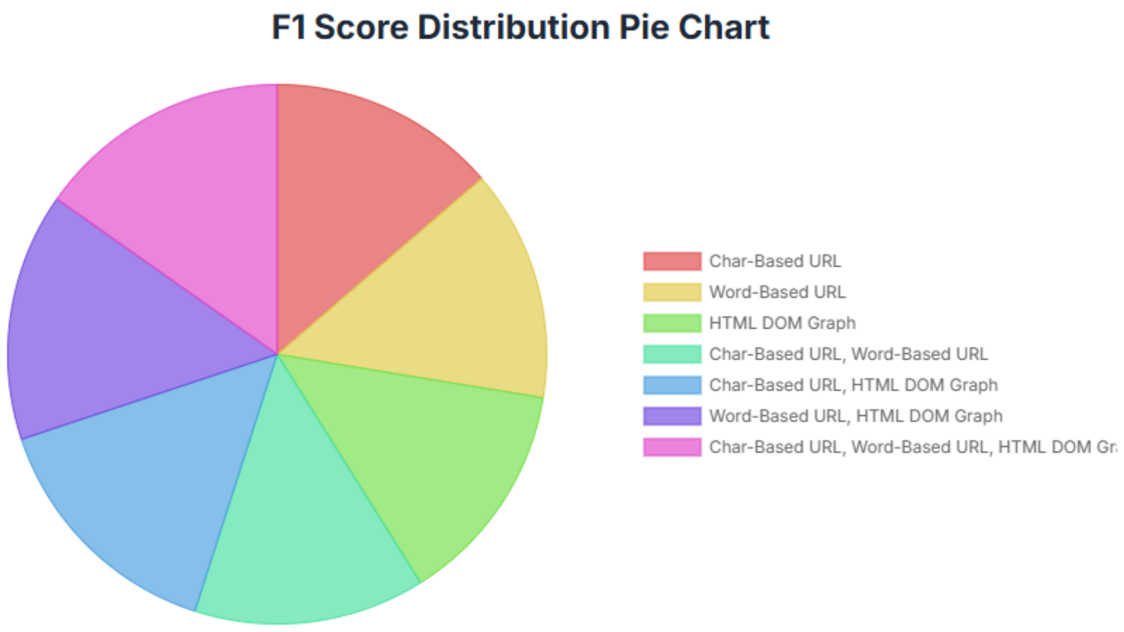


Real-world threat datasets are almost always imbalanced, and ignoring this can lead to ineffective or even dangerous models in practical deployments, especially for critical applications where false negatives (undetected phishing attacks) carry high risks. This indicates a commitment to robust experimental design, ensuring the validity and trustworthiness of the reported results. "In an effort to give back to the research community and further support advancements in phishing detection, we plan to release this dataset as an open-source benchmark" goes beyond merely reporting results. It signals a commitment to open science principles. By making their carefully curated and preprocessed dataset publicly available, the researchers not only provide transparency for their own work but also create a standardized resource for the broader research community. This means the researchers are fostering reproducibility, collaboration, and accelerated progress in the field of phishing detection. A common, high-quality benchmark dataset allows other researchers to directly compare their models under fair conditions, validate new approaches, and build upon existing work more effectively. This is a positive ethical and practical impact, contributing to a more robust and collaborative cybersecurity research ecosystem.

4.2. Implementation Details and Hyperparameter

Settings

The implementation of these experiments for DeepPhish-X utilized the Python deep learning library **PyTorch** (version 2.0.1), integrated 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. Experiments were conducted on an **NVIDIA A6000 GPU**. DeepPhish-X's CNN component, drawing on computer vision principles, employs two 2D convolutional layers with Dropout layers in between to prevent overfitting. The Transformer component, a key element of natural language processing, includes embedding, positional encoding, Dropout, and Transformer encoder layers to capture intricate patterns within the data. The GCN component, representing the graph neural network aspect, uses GCNConv layers and global average pooling to effectively model graph-structured HTML DOM data. The DeepPhish-X integrated model combines these components, utilizing an embedding layer followed by Transformer encoders and linear layers to combine the strengths of each individual model. This setup allows for the capture of diverse features and dependencies, leading to improved overall phishing detection performance. Specific hyperparameter settings, such as the number of units, activation functions, and number of parameters, were optimized to ensure effective model training and evaluation. Table 4 summarizes the layers and configurations used in the DeepPhish-X integrated model, providing a clear overview of the implementation details and parameter settings.



Convolutional Neural Network	Number of Parameters	Transformer	Number of Parameters	Graph Convolutional Network	Number 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 Average Pooling	–

Ensemble for selectively weighting and combining Transformer-based features	Operation	Number of Units/Heads	Activation Function	Number of Parameters
	Embedding	8	relu	40,000
	Positional Encoding	–	–	–
	Dropout	–	–	–
	Transformer Encoder	2 heads	relu	672

Ensemble for selectively weighting and combining Transformer-based features	Operation	Number of Units/Heads	Activation Function	Number of Parameters
	Linear	2	-	66

The explicit mention of "NVIDIA A6000 GPU" as the hardware for experiments, along with the detailed parameter counts in Table 4, clearly indicates that **DeepPhish-X is computationally intensive**. Training such a sophisticated hybrid deep learning architecture, which includes graph convolutions, requires substantial computational resources. This implies that while the model achieves high accuracy, its computational demands for training and potentially for real-time inference might present practical limitations. Deploying such a system in resource-constrained environments or scaling it to handle extremely high web traffic in real-time might necessitate dedicated hardware, distributed computing, or further model optimizations (e.g., quantization, pruning). This highlights a common trade-off in advanced deep learning solutions: higher accuracy often comes with higher computational costs, which is a critical consideration in practical cybersecurity applications.

4.3. Performance Comparison

This section evaluates the performance of DeepPhish-X against various baseline models and state-of-the-art techniques using **10-fold cross-validation**. The key evaluation metrics considered are **accuracy, precision, recall, and F1-score**.

Baseline Network Performance

Convolutional Neural Networks (CNNs) exhibit strong performance, indicating their effectiveness in capturing URL character-level features. Similarly, **Transformer models** perform well, demonstrating their ability to handle sequential dependencies in URLs. However, **Graph Convolutional Networks**

(GCNs) show slightly lower performance compared to CNNs and Transformers, which might be attributed to the inherent complexity of graph neural networks modeling HTML DOM trees.

Comparative Study Performance

In the comparative study, the **URLNet model** outperforms several other models, achieving significant accuracy and precision. The **Texception model**, while having high recall, exhibits notable variability in precision. **PhishDet**, on the other hand, achieves the highest scores across most metrics, confirming its robustness and reliability in phishing detection tasks.

DeepPhish-X (Proposed Ensemble Model) Performance

DeepPhish-X demonstrates exceptional performance, achieving a **22% improvement in precision** and a **23% improvement in recall** compared to the baseline models. These results underscore the superiority of this hybrid approach, which combines the strengths of CNN (computer vision), Transformer (natural language processing), and GCN (graph neural networks) to create a more comprehensive and effective phishing detection system. DeepPhish-X's ability to leverage multiple data modalities significantly enhances its detection accuracy and robustness against various phishing techniques. Table 5 provides a detailed comparison of performance metrics for all evaluated models, illustrating the effectiveness of this ensemble approach in improving phishing detection accuracy and reliability. Each metric's best performance is highlighted in bold.

Method	Accuracy	Precision	Recall	F1 Score
Baseline 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

Method	Accuracy	Precision	Recall	F1 Score
URLDet	0.8014 ± 0.0214	0.4007 ± 0.1042	0.5089 ± 0.1189	0.4959 ± 0.0510
HTMLDet	0.9154 ± 0.0292	0.8784 ± 0.0221	0.8416 ± 0.0301	0.8596 ± 0.0255
Comparative Studies				
URLNet	0.9425 ± 0.0208	0.9033 ± 0.0785	0.8051 ± 0.1195	0.8453 ± 0.0637
Texception	0.8534 ± 0.1308	0.8457 ± 0.1536	**0.9714 ± 0.0286**	0.8889 ± 0.0954
WebPhish	0.9290 ± 0.0719	0.9336 ± 0.0353	0.8689 ± 0.1311	0.8937 ± 0.1386
PhishDet	**0.9853 ± 0.0058**	**0.9522 ± 0.0204**	0.9721 ± 0.0070	**0.9620 ± 0.0091**
Our (DeepPhish-X)				
Our	0.9812 ± 0.0033	0.9658 ± 0.0125	0.9765 ± 0.0042	0.9709 ± 0.0050

4.4. Hyperparameter Impact Analysis

This section explores the impact of different hyperparameters on the performance of the DeepPhish-X phishing detection model. A crucial hyperparameter in the model is the **number of heads** used in the Multi-Head Attention mechanism, a core component of the Transformer inspired by natural language processing. The multi-head attention mechanism allows the model to concurrently attend to different parts of the input data, thereby providing a richer and more diverse representation. This study experimented with four different numbers of attention heads: 4, 8, 16, and 32, to determine the optimal configuration for DeepPhish-X. 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 F1-score of 0.9927. This suggests that using eight heads provides 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.

Number 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

This section presents the results of the ablation study to evaluate the importance...

4.9. Adversarial Attack Analysis

As shown by the results, the performance of DeepPhish-X degrades with increasing epsilon values. With no perturbation (ϵ=0), the model achieves 98.73% accuracy, with high macro-average precision, recall, and F1-score. However, even a small perturbation (ϵ=0.02) reduces the accuracy to approximately 97.32%, with a noticeable drop in performance, especially in the ability to correctly classify benign categories. As the perturbation becomes more significant (ϵ=0.04 and above), the model's accuracy further declines. At ϵ=0.08, the model's accuracy drops to 74.68%, and the confusion matrix indicates a strong bias towards misclassifying benign samples as phishing.

These findings highlight **DeepPhish-X's vulnerability to adversarial attacks**, especially when small but targeted perturbations are applied. This emphasizes the need to incorporate more robust defense mechanisms into the model, such as adversarial training or other forms of regularization, to enhance its resilience against such attacks.

4.10. Discussion: Case Study

This section analyzes specific cases to understand DeepPhish-X's performance, particularly focusing on instances where the model correctly and incorrectly classified URLs, as detailed in Table 10. This analysis provides insights into both the strengths and limitations of this approach.

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

Figure 6: Comparative HTML DOM Graph Visualizations for Benign and Phishing Case Studies

In Figure 6a, the URL "https://ninecloud.ae/our-services/interior-exterior-paint-contractors-in-dubai/" 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[1] with a coherent HTML structure, aligned well with the benign patterns the model had learned. In Figure 6b, the URL "https://pub-88f64e013ca94e82aa5d15393134722c.r2.dev/logs.html" was correctly classified as phishing. The DOM graph visualization, as shown in Figure 6b, reveals a complex and suspicious structure, and the URL itself presents. The presence of random characters and deceptive paths suggests phishing characteristics, which the model successfully identified. In Figure 6c, the URL "https://rilm.am/wp-content/uploads/2022/07/12e47ac82164e89a8c15f399384e6572.pdf" was incorrectly classified as phishing. The DOM graph visualization (Figure 6c) shows a complex structure, which may have misled the model. Despite being a benign URL, its intricate structure.