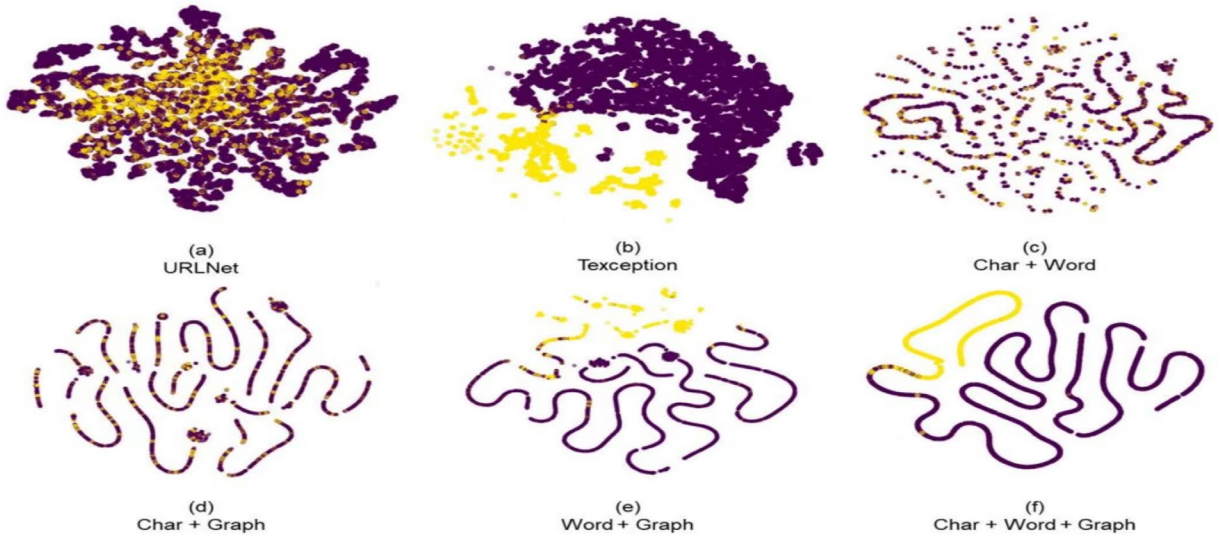
DeepPhish-X: Multi-Modal Feature Engineering for Phishing Detection Using Hybrid Models of Computer Vision, Natural Language Processing, and Graph Neural Networks

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):	

3. Proposed Method



This section introduces **DeepPhish-X**, a phishing detection framework that leverages Graph Convolutional Networks and Transformer Networks, integrating multi-modal features from HTML DOM Graphs and URL characteristics. As shown in Figure 3, DeepPhish-X combines different deep learning models to achieve a more accurate classification of phishing webpages.[1]

Figure 3: Overview of the DeepPhish-X phishing detection framework integrating HTML DOM graphs and URL features [1]

(Figure 3, showing the framework overview, should be inserted here)
DeepPhish-X is designed to integrate various features for detecting phishing attacks, while recognizing that URL, SSL certificates, HTML DOM, webpage content, HTML headers, and protocols, although available, are often susceptible to manipulation.[1] The core idea is that the most influential feature in determining whether a case is phishing varies from instance to instance.[1] Therefore, it is crucial to leverage as many features as possible simultaneously, while excluding those that might hinder learning due to manipulation, thereby maximizing phishing detection capabilities.[1]

3.1. URL and HTML Data Representation for Phishing Detection

To effectively detect phishing webpages, DeepPhish-X employs multi-modal feature engineering, analyzing and representing data from multiple perspectives. This section outlines the approach for representing both URL and HTML data, which is crucial for identifying patterns and characteristics unique to phishing attempts.[1] DeepPhish-X uses deep learning techniques to process and extract features from URLs and HTML content, providing a robust framework for phishing detection.[1]

3.1.1. Char-Based URL Feature Extraction (Computer Vision Inspired)

Character-based URL feature extraction in DeepPhish-X involves converting each URL into a matrix representation using one-hot encoding. Each character in a URL is mapped to its corresponding ASCII value, resulting in a numerical representation.[1] As outlined in Algorithm 1, the ASCII values are then padded or truncated to a fixed length of 128 characters to maintain uniformity across all URLs, and one-hot encoded into a 128x128 matrix.[1] This fixed-size input is suitable for CNN processing, drawing inspiration from

Computer Vision techniques that process grid-like data.

Algorithm 1: Char-based URL Representation for Phishing Detection [1]

1. Input: A URL string
2. Output: A 128x128 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] + * (max_length - len(ascii_values)))[:max_length]
9. End function
10. Function OneHotEncode(ascii_values)
11. matrix = zero matrix of size 128x128
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, DeepPhish-X proceeds 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 character sequences.[1] They are particularly effective for tasks involving spatial hierarchies and pattern recognition. In the context of char-based URL feature extraction, CNNs are used to capture local patterns and dependencies among characters in the one-hot encoded URL matrix.[1] By applying convolutional filters across the matrix, CNNs can identify and learn important character combinations and sequences that may indicate phishing attempts.[1] The convolutional operation is given by Equation (1):

$$Hl=f(\phi(Wl,X)+bl) \quad (1)$$ [1]

where Hl is the output feature map of layer l, Wl represents the convolutional filter weights, ϕ denotes the convolution operation, bl is the bias term, and f(·) is the activation function.[1] 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.[1]

To train the CNN component of DeepPhish-X, the loss function is defined as the categorical cross-entropy loss, which measures the discrepancy between the predicted probabilities and the true labels.[1] The loss function

LCNN is given by Equation (2):

$$LCNN=-N1\sum i=1N\sum c=1Cyi,clog(\sigma(WOHL(Xi)+bO)c) \quad (2)$$ [1]

where N is the number of URL samples, C is the number of classes, yi,c is the true label of the i-th sample for class c, Xi is the i-th input sample, HL(Xi) is the feature map obtained from the CNN's last layer LCNN for the i-th input char-based URL, WO and bO are the weights and biases of the output layer, respectively, σ is the softmax function, and pi,c is the predicted probability of the i-th input belonging to class c, computed as σ(WOHL(Xi)+bO)c.[1]

Padding or truncating URLs to a fixed length of 128 characters, and converting them into a 128x128 one-hot encoded matrix, is a common but crucial 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 like URLs. This design choice

implies a trade-off: while it enables efficient batch processing and model training, it might also lead to the loss of certain unique features in very long URLs due to truncation, or dilution of patterns in very short URLs due to padding. This could affect the model's performance on extreme URL lengths. This suggests that max_length (128 in this case) is a hyperparameter that balances the need to capture 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 with segmenting URLs into tokens using special characters such as slashes (/), dots (.), and hyphens (-) as delimiters.[1] These tokens are further refined by filtering out common stopwords and retaining only those with semantic significance.[1] Subsequently, a frequency analysis is performed on the tokens to construct a vocabulary dictionary containing the most frequently occurring 5000 words, each mapped to a unique integer identifier.[1] The tokenized URLs are transformed into sequences of integers and padded to a uniform 20-token length to facilitate batch processing.[1] These integer sequences serve as inputs to the Transformer network, a key component from

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.[1] The attention mechanism computes contextual relevance by evaluating the importance of each token within the sequence, enabling the model to effectively capture long-range dependencies.[1]

Algorithm 2: Word-based URL Representation for Phishing Detection [1]

1. Input: A list of URLs
2. Output: Transformed URLs into sequences of integers
3. Initialize 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 = if word in D else 0 for word in words]
15. Return (sequence[:20] + * 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 =

To effectively represent URLs as sequences of words, DeepPhish-X first splits each URL string using a set of predefined special characters. This splitting process converts the URL into a list of words. Next, a dictionary of the most frequently occurring words across the dataset is built, 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.[1] Algorithm 2 describes the detailed steps for converting URLs into word-based sequences of integers.[1]

Following this preprocessing, the transformed URL sequences are input into a Transformer Network to capture the contextual relationships and dependencies among the words.[1] Transformer Networks are particularly well-suited for this task, as they excel in modeling long-range dependencies and capturing intricate patterns within sequential data, a hallmark of

Natural Language Processing.[1] In the context of word-based URL feature extraction, DeepPhish-X utilizes the Transformer's ability to understand the contextual meaning of words within a URL by employing self-attention mechanisms.[1] The attention mechanism, as shown in Equation (3), computes the relevance of different parts of the sequence, allowing the model to dynamically weigh the importance of each token:

Attention(Q, K, V) = softmax(QK^T / sqrt(d_k))V (3) [1]

where Q, K, V are the query, key, and value matrices, and dk is the dimension of the key vectors.[1]

For a given input sequence X, the multi-head attention is defined as Equation (4):

MultiHead(Q,K,V) = Concat(head_1,...,head_h)W^O [1]

head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) (4) 1

where WiQ、WiK、WiV are learned projection matrices.[1] 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 mechanism.

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

FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 (5) [1]

where W1、W2、b1、b2 are learned parameters.[1] The final output from the Transformer model is given by Z = Transformer(X), representing the high-level features extracted from the input sequence.[1] The Transformer function is defined as Equation (6):

Transformer(X) = FFN (MultiHead(Q, K, V)) (6) [1]

These high-level features extracted by the Transformer are then fed into the final classification layer of DeepPhish-X to detect phishing webpages.[1] To train the Transformer component, the loss function is defined as the categorical cross-entropy loss, which measures the discrepancy between the predicted probabilities and the true labels.[1] The loss function

LTransformer is given by Equation (7):

LTransformer=−N1∑i=1N∑c=1Cyi,clog(σ(WOZi+bO)c) (7) [1]

where Xi represents the i-th input sequence, Zi is the Transformer's output for the i-th input sequence.[1]

WO and bO are the weights and biases of the output layer, respectively, σ is the softmax function, and pi,c is the predicted probability of the i-th input belonging to class c, computed as σ(WOZi+bO)c.[1]

DeepPhish-X simultaneously employs character-based (CNN, inspired by Computer Vision) and word-based (Transformer, from Natural Language Processing) URL analysis. Character-level analysis excels at detecting subtle misspellings, character insertions, or anomalous character sequences (e.g., "gooogle.com" or encoded characters), which are common phishing strategies. Conversely, word-based analysis focuses on the semantic meaning and contextual relationships of words (e.g., "login.php" appearing in an abnormal domain, or combinations like "secure-update-account"). This dual approach ensures DeepPhish-X can identify phishing attempts that exploit 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, particularly their multi-head attention mechanism, for word-based URL feature extraction is significant, as it excels at capturing "long-range dependencies" and "contextual relevance." Unlike simpler methods that merely look for suspicious keywords, the Transformer, a core **Natural Language Processing** model, can understand the relationships between words in a URL and dynamically weigh their importance within the sequence. For example, "secure-login.com" might be benign, but "login.secure-update.com" could be suspicious. The order and relationships of words are crucial. This indicates that DeepPhish-X aims to go beyond simple lexical feature matching. By understanding the contextual meaning of words in a URL, the Transformer can identify more subtle clues of malicious intent. This capability is vital for detecting sophisticated phishing URLs that might use legitimate words in deceptive orders or combinations, making the model more intelligent in interpreting URL semantics.

3.1.3. DOM Graph Feature Extraction (Graph Neural Networks)

HTML documents are parsed by DeepPhish-X using BeautifulSoup 4.12.3 (a Python 3.9 library for web scraping) to extract the DOM tree structure.[1] Each HTML element (e.g.,

, <a>, <p>) is represented as a node, and edges are established based on parent-child relationships in the DOM hierarchy.[1] This graph representation, a foundational element for

Graph Neural Networks, captures both the element types and their relational context, which is crucial for understanding the structural nuances of phishing pages.[1] The HTML content of each webpage is first parsed into a DOM tree using BeautifulSoup.[1] These elements' hierarchical relationships are preserved, with parent-child relationships represented as edges in the graph.[1] 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.[1] However, to maintain computational efficiency and focus on structural aspects, node features are limited to the tag names and simplified attribute representations, such as the presence or absence of key attributes (e.g., href, src).[1]

Algorithm 3: Graph-based HTML DOM Graph Representation for Phishing Detection [1]

1. Input: An HTML document
2. Output: HTML DOM tree's graph representation
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 return 0 # No nodes or edges exist
9. Initialize graph as an empty graph
10. Use a queue to perform level-order traversal of dom_tree
11. While queue is not empty do
12. node = dequeue()
13. Add node to graph as a vertex
14. For each child of node do
15. Add child to graph as a vertex
16. Add an edge from node to child in graph
17. End for
18. End while
19. Return graph
20. End function
21. dom_tree = ParseHTML(HTML document)
22. graph = BuildGraph(dom_tree)

This process begins with parsing the HTML document to construct the DOM tree.[1] This tree structure is then converted into a graph, where nodes represent HTML elements, and edges represent parent-child relationships between these elements.[1] Algorithm 3 provides the detailed steps for this transformation.[1] Once the DOM tree is represented as a graph, DeepPhish-X uses Graph Convolutional Networks (GCNs), a key component of

Graph Neural Networks, for feature extraction.[1] GCNs are well-suited for this task as they excel in capturing the relational and structural information inherent in graph data.[1] The following formulas outline the operations involved in applying GCNs to the graph representation of the HTML DOM graph.

In the context of DOM tree feature extraction, Graph Convolutional Networks (GCNs) extend the concept of convolutional networks to graph-structured data.[1] GCNs effectively learn node representations by aggregating features from neighboring nodes.[1] 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.[1] The graph convolutional operation is given by Equation (8):

$$H(I+1)=ReLU(D^{-21}A^D^{-21}H(l)W(l)) \quad (8)$$ [1]

where A^=A+I is the adjacency matrix with added self-connections, D^ is the diagonal node degree matrix of A^, H(l) is the matrix of activations in 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.[1] The initial input to the GCN,

H(0), is the feature matrix X derived from node attributes.[1] The final GCN layer

H(L)'s output serves as the high-level feature representation of the graph.[1]

To train the GCN component of DeepPhish-X, the categorical cross-entropy loss function is used, which measures the discrepancy between the predicted class probabilities and the true labels.[1] The loss function

LGCN is defined as Equation (9):

LGCN=−N1∑i=1N∑c=1Cyi,clog(σ(WOH(L)(Xi)+bO)c) (9) [1]

where N is the number of URL samples, C is the number of classes, yi,c is the true label of the i-th sample for class c, σ is the softmax function, Xi is the i-th input graph sample, and H(L)(Xi) is the output from the last GCN layer for the i-th input graph.[1] The use of GCNs significantly enhances phishing detection accuracy by effectively leveraging the structural information inherent in HTML DOM graphs.[1]

The core premise of DeepPhish-X is that phishing websites, while they may mimic legitimate URLs, rarely fully replicate HTML structure. By converting this structure into a graph and processing it with GCNs (a **Graph Neural Network** technique), the model essentially learns the webpage's "structural fingerprint." Legitimate websites typically have organized, complex, and consistent DOM structures, while phishing sites (often hastily created or templated) may exhibit simpler, irregular, or anomalous structural patterns. This underlying architectural difference is much harder for attackers to perfectly mimic or obfuscate than superficial URL changes. This suggests that the way a webpage is constructed (its underlying architecture and the relationships between elements) provides a more robust, difficult-to-forge 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 entirely 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 in capturing the relational and structural information inherent in graph data." This highlights a deliberate and intelligent selection of a deep learning architecture specifically

tailored to the intrinsic topological structure of the data. This suggests the increasing sophistication of applying deep learning to cybersecurity problems. It's not just about throwing generic neural networks at a problem, but recognizing that effective feature extraction often requires models designed specifically for the underlying data structure. This approach allows DeepPhish-X to leverage the 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. Ensemble Classifier Utilizing Char-Based URL, Word-Based URL, and HTML DOM Graph Features (Hybrid Model)

This section proposes **DeepPhish-X**, a multi-modal 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] However, many of these features can be manipulated to appear legitimate, potentially deceiving both users and phishing detection systems.[1] As stated by Wang et al., the key is that the most influential feature in determining whether a case is phishing varies for each instance.[1] Therefore, it is crucial to leverage as many features as possible simultaneously, while excluding those that might hinder learning due to manipulation, thereby maximizing phishing detection capabilities.[1]

DeepPhish-X introduces an ensemble classification model that integrates the strengths of CNNs (for **Computer Vision**-inspired char-based URL features), Transformers (for **Natural Language Processing**-based word-based URL features), and GCNs (for **Graph Neural Network**-based HTML DOM features).[1] 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.[1] The architecture of this hybrid ensemble model is illustrated in Figure 4.[1]

Figure 4: The DeepPhish-X ensemble classification model combining CNN (Computer Vision), Transformer (Natural Language Processing), and GCN (Graph Neural Network) components for phishing detection [1]

(Figure 4, showing the ensemble model architecture, should be inserted here)
The architecture of DeepPhish-X is designed to effectively integrate multiple data modalities for phishing detection, combining URL and HTML DOM features.[1] The selection of model components and hyperparameters was guided by empirical experimentation and best practices in the field of deep learning.[1] 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.[1] The architecture includes two convolutional layers with kernel sizes of 3x3, chosen for their ability to detect small patterns and variations in character sequences.[1] The ReLU activation function is applied to introduce non-linearity, and max-pooling layers are used to reduce dimensionality and computational cost.[1]

DeepPhish-X utilizes Transformer networks for word-based URL analysis, as they demonstrate superior ability to model long-range dependencies and context within sequences, a key strength in **Natural Language Processing**.[1] The model includes two attention heads, providing a balance between computational efficiency and the ability to capture complex relationships within word sequences.[1] Positional encoding is crucial for maintaining sequence order, enhancing the model's understanding of contextual word relationships.[1]

DeepPhish-X applies Graph Convolutional Networks (GCNs) to capture the relational information inherent in the HTML DOM structure, a fundamental application of **Graph Neural Networks**.[1] The architecture consists of two GCN layers to effectively aggregate information from neighboring nodes, with the number of features per node optimized to ensure a good trade-off between model complexity and accuracy.[1] The ReLU activation function, consistent with that used in CNNs, is utilized to ensure efficient learning across layers.[1]

The DeepPhish-X ensemble classifier consists of three primary components.[1] First, the CNN component processes the one-hot encoded URL character matrix, extracting local patterns and dependencies among characters.[1] The output is defined as

Hchar=CNN(Xchar).[1] Second, the Transformer component transforms URLs into sequences of integers based on word tokens, and captures long-range dependencies and contextual relationships among words.[1] The output is defined as

Hword=Transformer(Xword).[1] Third, the GCN component models the structural and relational information of the HTML DOM tree using graph convolutional operations.[1] The output is defined as

Hgraph=GCN(A,Xgraph).[1]

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

Hconcat=Concat(Hchar,Hword,Hgraph) [1]

Z=WfcTransformer(Hconcat)+bfc 1

P=softmax(Z) (10) 1
where Wfc and bfc are the weights and biases of the fully connected layer, Hconcat is the concatenated feature vector, and P represents the predicted probabilities for each class.[1]

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

LCNN=−N1∑i=1N∑c=1Cyi,clog(Pi,c) (11) [1]

where N is the number of URL samples, C is the number of classes, yi,c is the true label of the i-th sample for class c, and Pi,c is the predicted probability of the i-th sample belonging to class c.[1]

By integrating character-based URL (Computer Vision), word-based URL (Natural Language Processing), and HTML DOM graph (Graph Neural Network) features, and passing them through a Transformer layer, DeepPhish-X leverages the unique strengths of each feature representation, resulting in a more accurate and robust phishing detection system.[1]

The DeepPhish-X model's architecture is not simply a single, large deep learning model, but explicitly leverages an "ensemble" of specialized components: CNN for character-level patterns (Computer Vision), Transformer for word-level dependencies (Natural Language Processing), and GCN for graph-based structural patterns (Graph Neural Networks). This suggests that in complex domains like phishing, no single deep learning architecture is universally optimal for all types of features. Each component acts as an "expert" in extracting features from a specific data modality. This design philosophy indicates that for multifaceted and complex threats like phishing, where attackers employ diverse strategies, a holistic, ensemble approach combining the strengths of various specialized models is superior. This "ensemble of experts" paradigm, coupled with a sophisticated integration mechanism (the final Transformer layer), allows the system to capture a broader range of malicious indicators, making it more robust and accurate than any single-modality or single-architecture solution. This points to an important trend in advanced AI system design: building heterogeneous, modular architectures that leverage different neural network types' unique strengths.

In the final integration step of DeepPhish-X, applying a Transformer layer to the concatenated feature vector from the CNN (Computer Vision), word-based Transformer (Natural Language Processing), and GCN (Graph Neural Network) is a crucial architectural decision. While Transformers are known for their capabilities with sequential data (like text), applying them

here to integrate disparate feature types (character-level, word-level, and graph-based) highlights their versatility. The attention mechanism within the Transformer enables it to learn the interdependencies and relative importance among these diverse feature vectors, regardless of their original modality. This suggests that Transformer networks are powerful multi-modal fusion tools, extending beyond their initial applications in Natural Language Processing. Their ability to "selectively attend" to and weigh information from different sources allows them to identify subtle relationships and complementary information between otherwise disparate feature sets. This strategic use of the Transformer for integration is a key factor in DeepPhish-X's enhanced robustness and accuracy, allowing it to synthesize a comprehensive understanding of a webpage's malicious intent from multiple perspectives.